INF 504 Data Mining & Machine Learning
Lecture Notes

February 5, 2019

Contents

List of Tables

List of Figures

Lecture 1

Introduction

Contents

Main Objectives:

- Become familiar with the jargon of machine learning by comparing machine learning to statistics

$\mathcal{O}$

Reading Assignment:

- Many of the sources that I will use throughout the semester are cited in the next bullet. See the corresponding list of references at the end of this document. They are also listed in the course syllabus.
- [Bis06], [Bre01], [Don17], [GCS+14], [HTF01], [JWHT14], [KNNL05], [Mur12] , [RW06], [Wak13]
- You are expected to be familiar with regression and ANOVA as presented in, e.g., my course notes for INF 511 Modern Regression I. These notes are available to you in BbLearn as **INF511Notes.pdf**.

$\mathcal{R}$

1.1 Introduction to Machine Learning via Regression

- Below, we compare traditional statistical regression, which you might encounter in "STAT 102" and which should be familiar to you, to machine learning (ML).
- You should see that there is much overlap between Statistics and ML.
- In many respects, (supervised) ML takes to an extreme the traditional regression objective to **predict**, while traditional inferential procedures, such as testing or interval estimation, play little to no role in ML.
- The current introductory context, in this section, is that of **supervised learning**, the main focus of our course. We will get to defining this and other terms as we go.

1.1.1 Traditional Statistical Regression Perspective

Observed data:

$$(y_i, \boldsymbol{x}_i), \quad i = 1, \ldots, n$$

- y_i is a scalar (1-vector) continuous or discrete **quantitative** or **numerical** variable or (in the case of classification) a numerical code representing the non-numerical value of **categorical** or **qualitative** (either **nominal** or **ordinal**) variable.
- $\boldsymbol{x}_i$ is a p-vector of continuous or discrete numerical values, perhaps numerical codes representing non-numerical categorical variables.
- More terminology as we go.

A very popular (but, by no means, exhaustive) data model: **additive error data model**:

$$y = \mu(\boldsymbol{x}, \boldsymbol{\beta}) + \epsilon,$$

where

- ϵ is an **error** with mean, $\mathrm{E}(\epsilon) = 0$,
- so that $\mathrm{E}(y) = \mu(\boldsymbol{x}, \boldsymbol{\beta})$ is the **mean** model of y as a function of covariates and parameters, sometimes abbreviated to $\mu(\boldsymbol{x})$ or μ,
- $Var(\epsilon) = \sigma^2$
- so that $Var(y) = \sigma^2$,
- and $\boldsymbol{\beta}$ is a q-vector of unknown **parameters**, though some methods have no firm notion of parameters. Most often, $p = q$.
- ϵ typically includes (implicitly, at least) **measurement error** as well as error in the specification of the mean model μ, sometimes called **model bias** or **process (model) error**

- By no means does above additive model cover all situations, but it does cover the case where y is normal, which is important for our course.
- We're merely in an introductory mode for now.

- y is often called a **response** (variable) or, less often, **dependent variable**.

- And, $\boldsymbol{x}$, or its elements, is/are often called a **covariate**, **independent variable**, **predictor** (variable), **regressor**, **factor**, **grouping variable**, **control variable**, **dummy variable**, or **indicator variable**, the particular terminology often depending on the context in which the model is used.

Additive error model model in **matrix notation**:

$$\boldsymbol{y} = \boldsymbol{\mu} + \boldsymbol{\epsilon},$$

where

- $\boldsymbol{y}$ is the n-vector with y_i as its ith element, and
- $\boldsymbol{\mu}$ is the n-vector with $\mu(\boldsymbol{x}_i, \boldsymbol{\beta})$ as its ith element; again, the mean of $\boldsymbol{y}$ (evaluated at corresponding values of covariates and parameters).

- A **common objective in regression** is to estimate the mean of one or more unobserved or "new" or "future" y given a corresponding "new" observed covariate $\boldsymbol{x}$.
- That is, we often want to "**predict y at $\boldsymbol{x}$**," though most statisticians might conventionally say, "estimate the mean of y at $\boldsymbol{x}$."
- That is, **find a good "predictor"** (estimate/estimator of the mean),

$$\widehat{\mu} \equiv \mu(\boldsymbol{x}, \widehat{\boldsymbol{\beta}}).$$

- Statisticians often distinguish between **prediction** of a random variable y, versus **estimation** of the (fixed) mean of y, $E(y)$.

- ML (see next section) very often indiscriminately uses **prediction** for either case, usually referring, however, to estimation of the (fixed) mean of y.

- Upon obtaining an estimate (e.g., least squares, maximum likelihood), $\widehat{\boldsymbol{\beta}}$, we often hear such thing like, we have **fit the model to the data**,
- and we often refer to $\widehat{\boldsymbol{\beta}}$ as the estimate/estimator of the parameter, $\boldsymbol{\beta}$,
- and to $\widehat{\mu} \equiv \mu(\boldsymbol{x}, \widehat{\beta})$ as the **fitted mean model or function** or estimate/estimator of (the mean of) y (at $\boldsymbol{x}$).
- Statisticians usually estimate/fit the variance model, $Var(\epsilon) = \sigma^2$, too.

In addition to the objective of using $\widehat{\mu}(\boldsymbol{x})$ to "predict" y at one or more new observed covariates, $\boldsymbol{x}$, we might also use formal statistical inference procedures to estimate or test various functions of $\boldsymbol{\beta}$, e.g. **t or F based procedures**. This second objective of regression is usually not the focus of ML (see next section).

1.1.2 Machine Learning Perspective of Regression

Now, we review the above section using terminology of machine learning (ML), though we retain much of the above notation, which, in machine learning, often differs from the previous section.

Incidentally, again, we are using **supervised learning** in an introductory way for the moment. More, shortly.

- Covariate, $\boldsymbol{x}$, is often referred to as **input** or **feature** or **attribute**.
- Response, y, is often called **output** or **target** or **label**.
- The parameter, $\boldsymbol{\beta}$, is often called **weight** vector or, its elements, **weights**.
- See Figure 1.1 for an illustration of the additive error model.

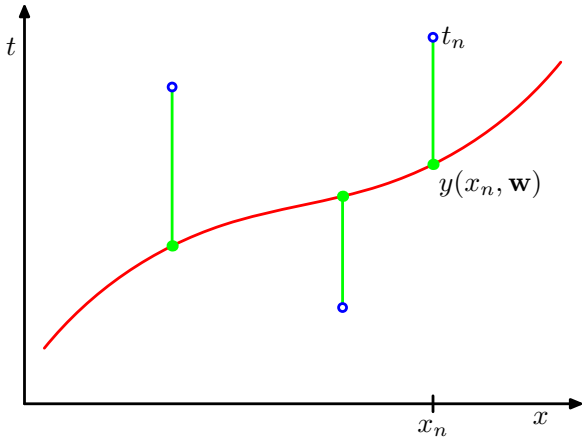


Figure 1.1: Illustration of additive error model (Source: [Bis06]). Translation to our notation: y=t, $y_n = t_n$, $\mu(\boldsymbol{x}_n, \boldsymbol{\beta}) = y(x_n, \boldsymbol{w})$.

- **ML Goal:** y. But, y may be relatively difficult to obtain.
- Instead, $\boldsymbol{x}$ is relatively easy to observe, and we hope it can tell us about y.
- E.g., you may have many images of people (or other things) ($\boldsymbol{x}$), but you do not know the identity, y, of people (facial recognition/identification).

- Of course, we envision a relationship between $\boldsymbol{x}$ and y, and we want to **learn** about that relationship in the hope of using $\boldsymbol{x}$ to tell us something about y. Thus, we begin to understand the "input-output" terminology.
- We use observed ("past") y values to **teach** us about or **supervise our learning** of the relationship between itself and $\boldsymbol{x}$,
- so that we may use the learned relationship to let other, "future" observed inputs, $\boldsymbol{x}$, help us reach our goal in the form of a (hopefully good) prediction of the y that we imagine is associated with $\mathbf{x}$, but did not observe with $\mathbf{x}$.

- A bit more formally, we characterize the unknown relationship to be learned as $y \approx \mu(\boldsymbol{x}, \boldsymbol{\beta})$ (if y has error), where the form of $\mu(\boldsymbol{x}, \boldsymbol{\beta})$, including $\boldsymbol{\beta}$, is unknown.
- Thus, we use observed data, i.e., observed inputs and outputs $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$ to **learn** about (i.e., fit/estimate) the function(s), $\mu(\boldsymbol{x}, \boldsymbol{\beta})$, that relates inputs to outputs.
- The data set, $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$ is often called the **training set** or **model-building set** because, well, it is the set used to "train" or "build" or fit the (one or more) model(s). More later on other sets: **validation set** and **test set**.
- With additional, observed, "future" inputs, $\boldsymbol{x}$, we use the learned relationship, to tell use about (predict) "future" outputs, i.e., $y \approx \widehat{\mu} \equiv \mu(\boldsymbol{x}, \widehat{\boldsymbol{\beta}})$

- **Prediction** is sometimes referred to as **generalization** because we

learn from an observed (training) data set, but wish to predict (generalize) to other values not in the observed data set.

- In a slightly different context, generalization may be referred to as **test set performance**, performance of prediction for points in a test set, which, as we may learn, is different from the training set, hence the term "generalization" (beyond the training set).
- ML does not typically use distinguishing terminology for prediction of a random variable, y, versus estimation of its (assumed fixed) mean, $E(y) = \mu$. More often than not, in ML, "prediction" means estimation of the mean model/function, μ, not prediction of a random y.
- The estimated/fitted function, $\widehat{\mu} \equiv \mu(\boldsymbol{x}, \widehat{\boldsymbol{\beta}})$, is sometimes referred to as a **learner**.
- More generally, a **learner** is a **procedure** or **algorithm** for (finding and) estimating relationships.
- Thus, **regression** is an example of **supervised learning** or **learning with a teacher**.
- Because we live in 2019, practically all of the learning procedures are (semi-)automated, performed by machines, and we have the terminology, **(supervised) machine learning**.
- Thus, you (should) already know a lot about ML, with much of the difference between ML and Statistics being terminology!

- As mentioned above, much of **traditional regression** also concerns other inference procedures, such as **hypothesis testing** or **interval estimation** of functions of the mean or its parameters, but **ML** typically does not focus on such procedures. Thus, it is the **focus**

on prediction that tends to distinguish (supervised) ML from more traditional regression methods. Discussion in class.

- And, in ML, we are not usually interested in $\boldsymbol{\beta}$ or, more generally, we are not usually interested in the particular form of $\mu(\boldsymbol{x}, \boldsymbol{\beta})$ used to get at unobserved y. We usually don't belabor estimates of functions of the mean or hypothesis tests about functions of the mean.
- Also, we have tended to present the above discussion in a semi-probabilistic way, as you might see in a statistics class (random error, expectations, variances, etc.). There are alternative approaches to estimating a function between observed quantities (e.g., function approximation methods).
- See `https://statweb.stanford.edu/~tibs/stat315a/glossary.pdf` for a somewhat humorous comparison of ML and Statistics.
- [Bre01] characterizes two cultures: **data (statistical)** and **algorithmic (computational)**, where the latter alludes to ML's deemphasis of the particular form of μ; in some sense, ML simply seeks a procedure or algorithm that predicts well.
- More recently, inasmuchas ML or Statistics has morphed into "data science", the dichotomy has been summarized as **prediction** vs **inference** ([Don17]).

Example: **Non-parametric regression** using a **Gaussian process** (GP).

- We save details for later but simply say here that GPs offer flexible curves for fitting complex data for which a more traditional parametric model might escape us.
- Figure 1.2 shows atmospheric CO_2 concentrations (ppms) measured atop Mauna Loa along with a (Bayesian) predictive confidence region.

- Figure 1.3 shows three component GPs fit to the data: a long-term GP, a short-term GP, and a seasonal GP. The sum of these, superimposed on the previous figure, would nicely illustrate our estimated (Bayesian posterior mean) function, $\widehat{\mu}$, for these data.
- Figure 1.4 shows a contour plot of the seasonal GP component "wrapping upwards over years." Interestingly, perhaps, you see the May peak lessen (prediction beyond 2003 indicated by the blue dashed line), and the October trough appears to deepen slightly.

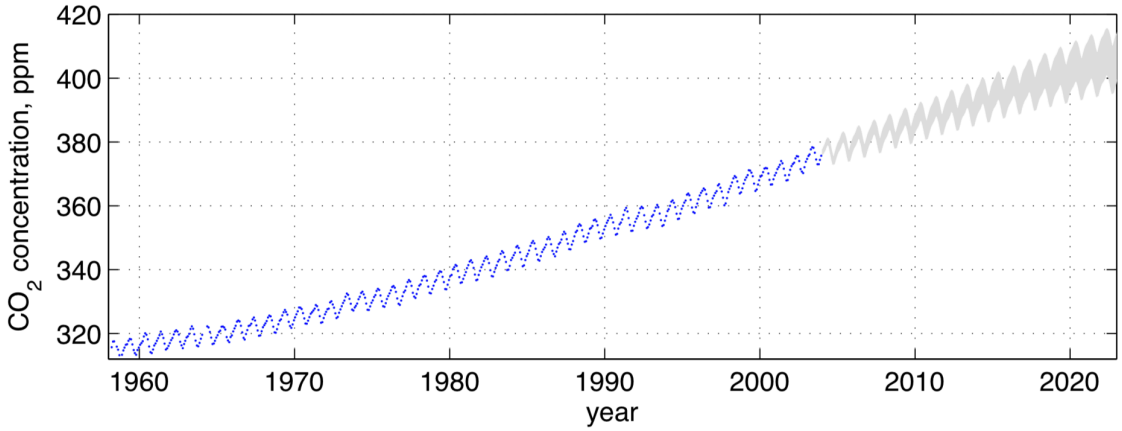


Figure 5.6: The 545 observations of monthly averages of the atmospheric concentration of CO_2 made between 1958 and the end of 2003, together with 95% predictive confidence region for a Gaussian process regression model, 20 years into the future. Rising trend and seasonal variations are clearly visible. Note also that the confidence interval gets wider the further the predictions are extrapolated.

Figure 1.2: (Source: [RW06])

.

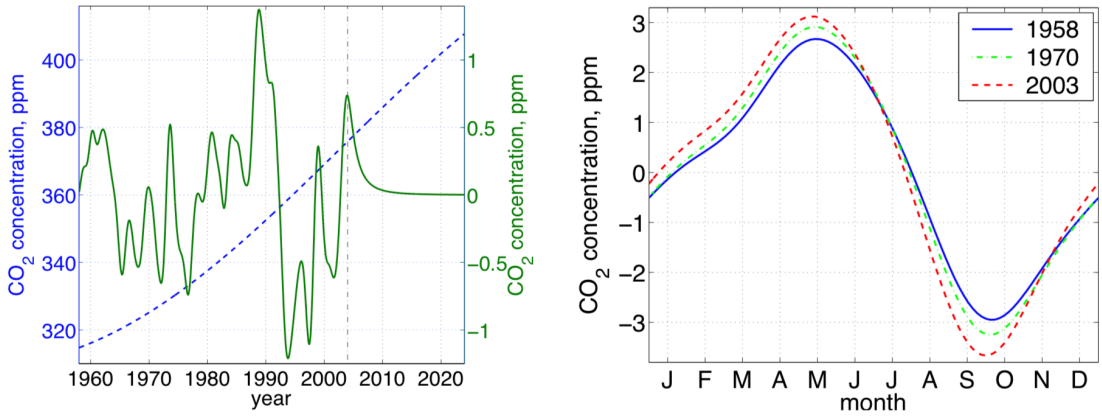


Figure 5.7: Panel (a): long term trend, dashed, left hand scale, predicted by the squared exponential contribution; superimposed is the medium term trend, full line, right hand scale, predicted by the rational quadratic contribution; the vertical dash-dotted line indicates the upper limit of the training data. Panel (b) shows the seasonal variation over the year for three different years. The concentration peaks in mid May and has a low in the beginning of October. The seasonal variation is smooth, but not of exactly sinusoidal shape. The peak-to-peak amplitude increases from about 5.5 ppm in 1958 to about 7 ppm in 2003, but the shape does not change very much. The characteristic decay length of the periodic component is inferred to be 90 years, so the seasonal trend changes rather slowly, as also suggested by the gradual progression between the three years shown.

Figure 1.3: (Source: [RW06])

.

1.2 Classification

- In the previous sections, you may have been thinking about regression as relating a quantitive y to covariates $\boldsymbol{x}$, but much of the discussion is equally applicable to y that assume non-numerical or **categorical** or **qualitative** values (**nominal** or **ordinal** (y takes on values of numerical codes representing values of non-numerical categorical variable)).
- Instead of saying that we wish to "**regress y on x**", we might say that we want to "**classify y given x**."

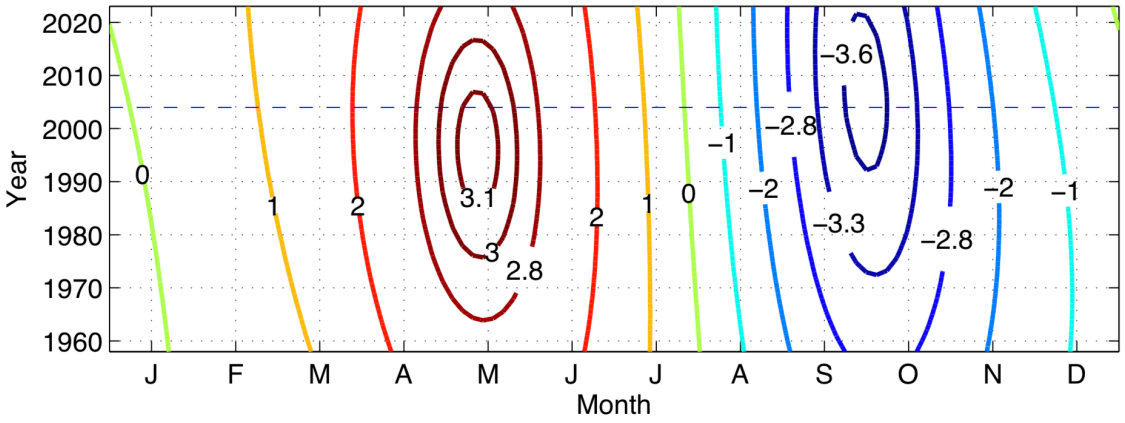


Figure 5.8: The time course of the seasonal effect, plotted in a months vs. year plot (with wrap-around continuity between the edges). The labels on the contours are in ppmv of CO_2. The training period extends up to the dashed line. Note the slow development: the height of the May peak may have started to recede, but the low in October may currently (2005) be deepening further. The seasonal effects from three particular years were also plotted in Figure 5.7(b).

Figure 1.4: (Source: [RW06])

.

- Sometimes called **pattern recognition** or **image classification** (if your inputs are pixel intensities).
- Thus, classification and regression are two sides of the same (supervised learning) coin.
- See Figures 1.5 and 1.6.

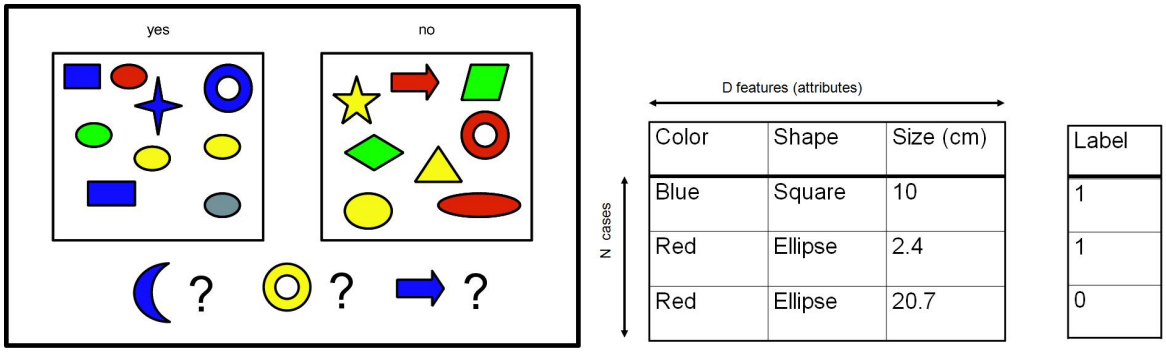


Figure 1.5: Toy illustration of classification (Source: [Mur12])

.

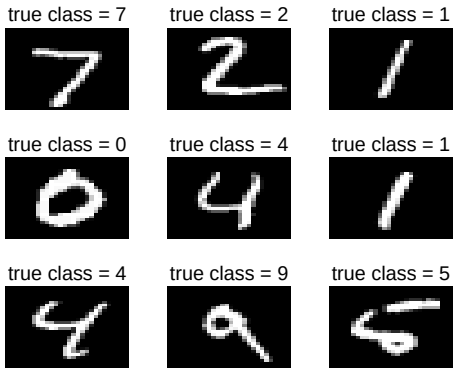


Figure 1.6: Image classification (Source: [Mur12])

.

1.3 Unsupervised Learning

- In unsupervised learning, in some sense, we know relatively little compared to supervised learning. A fortiori, we still wish to learn.
- In particular, we typically **do not have a notion of "response" and "covariate" or of "output" and "input."**
- In this case, we typically have a variable (vector) $\boldsymbol{x}$, without distinguishing any of its elements as "inputs" or "outputs". (Though, we

may use supervised methods by treating some elements as inputs and others as outputs.)

- In other words, **we do not have outputs to teach us about or supervise our learning of a relationship between inputs and outputs.**
- We also may say that we are **learning without a teacher**.
- So, we use our observed data $\boldsymbol{x}_i$, $i = 1, \ldots, n$, to **discover** relationships or patterns among the variables in $\boldsymbol{x}$ or to simply **describe** or **visualize** the data. Upon discovery of relationships, then, perhaps, we use supervised (regression/classification) methods.
- Sometimes called **knowledge discovery**.

1.3.1 Clustering (Data Segmentation) & Density Estimation

- The basic idea of **clustering** is to discover K groupings, i.e., **clusters**, of the data points, $\boldsymbol{x}_i$, $i = 1 \ldots, n$.
- In other words, **segment** the data into K groups.
- Clustering is sometimes referred to as **data segmentation** (or **image segmentation** if your data points consist of e.g., pixel intensity values).

Example: The K**-means** clustering algorithm is simple:

(0) Assume there are K clusters and create an initial collection of K clusters of points (an initial set of point sets).

(1) Compute the K means (averages), one for each cluster.

(2) Create K *new* clusters by reassigning points to the closest of the *current* K means computed from the *current* K clusters in (1).

(3) Repeat (1) and (2) until clusters assignments to not change.

- See, e.g., [HTF01, Alg. 14.1] [JWHT14, Alg. 10.1], or [Bis06, §9.1].
- See Figure 1.7 which illustrates the K-means ($K = 2$) algorithm in 2D, i.e., the points, $\boldsymbol{x}_i$, are 2-vectors $\boldsymbol{x}_i = (x_{i1}, x_{i2})^t$ of (standardized) Old Faithful eruption duration x_{i1} and time to next eruption, x_{i2}. (K and D are only coincidentally the same in this simple example.)

Example: Clustering via density estimation using **finite Gaussian mixture models**

- Basic idea is to assume each data point, $\mathbf{x}_i$, came from one of K multivariate normal distributions (but, in an unsupervised scenario, we don't know which distribution a point belongs to).
- The multivariate density of the variables, $\mathbf{x}_i$ can then be expressed as a probability mixture of normal (Gaussian) distributions, with unknown means, (co)variances, and probabilities.
- Thus, e.g., use maximum likelihood (or Bayesian methods) to estimate the mean and covariance parameters (and K probabilities of belonging to each of the K distributions).
- Figure 1.8 shows an initial and a near-final step of the EM algorithm (for implementing maximum likelihood; details later perhaps) for the same Old Faithful data used in the above K-means clustering example and Figure 1.7, above. Points are colored as a weighted average of red and blue (in the sense of some color model...) where weights correspond to the probability of the points having been generated by one (blue) or the other (red) Gaussian mixture component.

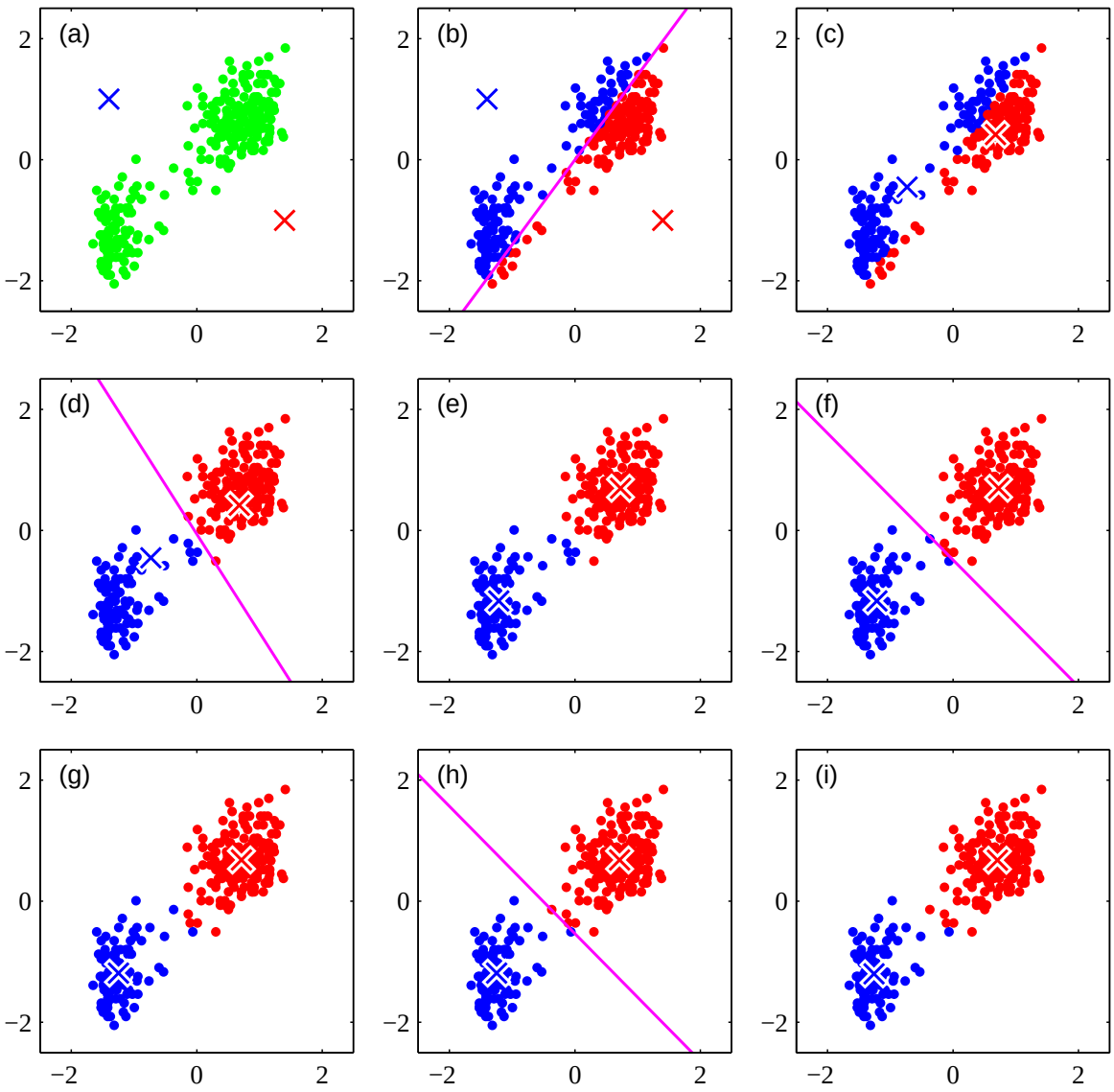


Figure 1.7: Graphical Illustration of the K-means clustering algorithm. (a-b) Step (0); (c-d) Steps (1)-(2); (e-f) Steps (1)-(2); (g-h) Steps (1)-(2); (i) Step (1) and stop because means hence clusters (would-be Step (2)) are the same as last step (h). (Old Faithful time to next eruption (vert.) vs. current eruption duration (horiz.) (standardized).) (Source: [Bis06])

Example: Clustering via **Bayesian non-parametric density estimation** by infinite mixture models using a **Dirichlet process** (DP) prior.

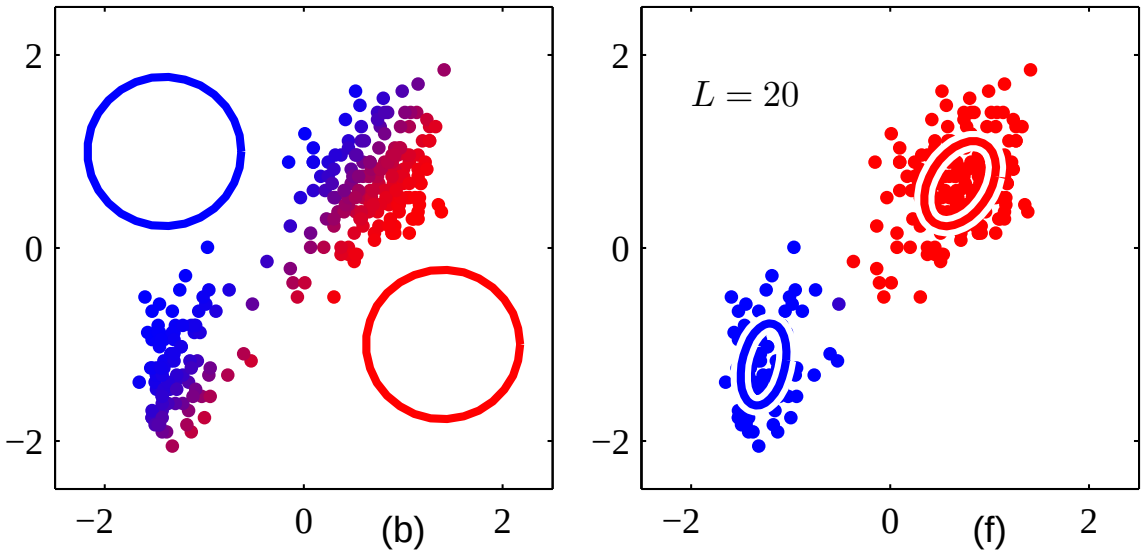


Figure 1.8: Graphical depiction of the $K = 2$ Gaussian mixture model (pdf) fit to the Old Faithful data (time to next eruption (vert.) vs. current eruption duration (horiz.) (standardized)). Ellipses depict 1 standard deviation contours of the $K = 2$ mixture components (the 2 Gaussian distributions) (Source: [Bis06]).

- In short, unlike the finite ($K < \infty$) mixture of component Gaussian models, we model a probability density of points as (countably) infinite mixture of component densities where the component densities are not a priori specified to be Gaussian but are given a prior distribution (yes, a (prior) distribution of distributions).
- A **Dirichlet process (DP)** is such a (prior) distribution for distributions (of a continuous random variable).
- DPs are analogous to the **Dirichlet probability density functions (pdfs)** which are often used as a (prior) distribution for the probability masses of discrete distributions, i.e. for the probabilities associated with random variables that can only assume a finite number of possible values.
- E.g., random variable $Y \sim binomial(n, p)$ can assume only one of two (finite!) possible values, often labeled as "failure" and "success,"

and encoded numerically as 0 and 1. We often denote $p = P(Y = 1)$ (probability of "success"). A beta pdf, a special case of Dirichlet pdf, is often used as a (prior) distribution on the two (finite!) probabilities, $p_1 = p$, $p_2 = (1-p)$. That is, this Dirichlet distribution is a (degenerate bivariate) pdf for the two random variables, p_1 and $p_2 = (1 - p_1)$ such that $p_1 \geq 0$, $p_2 \geq 0$ and $p_1 + p_2 = 1$ (thus, $p_1 = 1 - p_2$, $p_1 \leq 1$ and $p_2 \leq 1$).

- We leave details of DPs (and perhaps of Dirichlet pdfs) for later, time permiting.
- Figure 1.9 shows three Gibbs samples from the posterior of a Bayesian mixture model, using a DP prior, along with the posterior probabilities of the number k of components. Notice that only a finite number of components are sampled for intergers k with relatively high posterior probabilities. (Not the Old Faithful data!)

1.4 Other Topics

We will also cover other topics, including but not necessarily limited to

- Linear model variable/model selection, including information criteria, model (generalized cross-) validation ((G)CV), sparse modeling, shrinkage, regularization, LASSO, ridge regression, penalized likelihood, Bayesian model comparison/selection
- Neural networks
- Support vector machines (SVMs)
- Relevance vector machines (RVMs)
- Kernel PCA

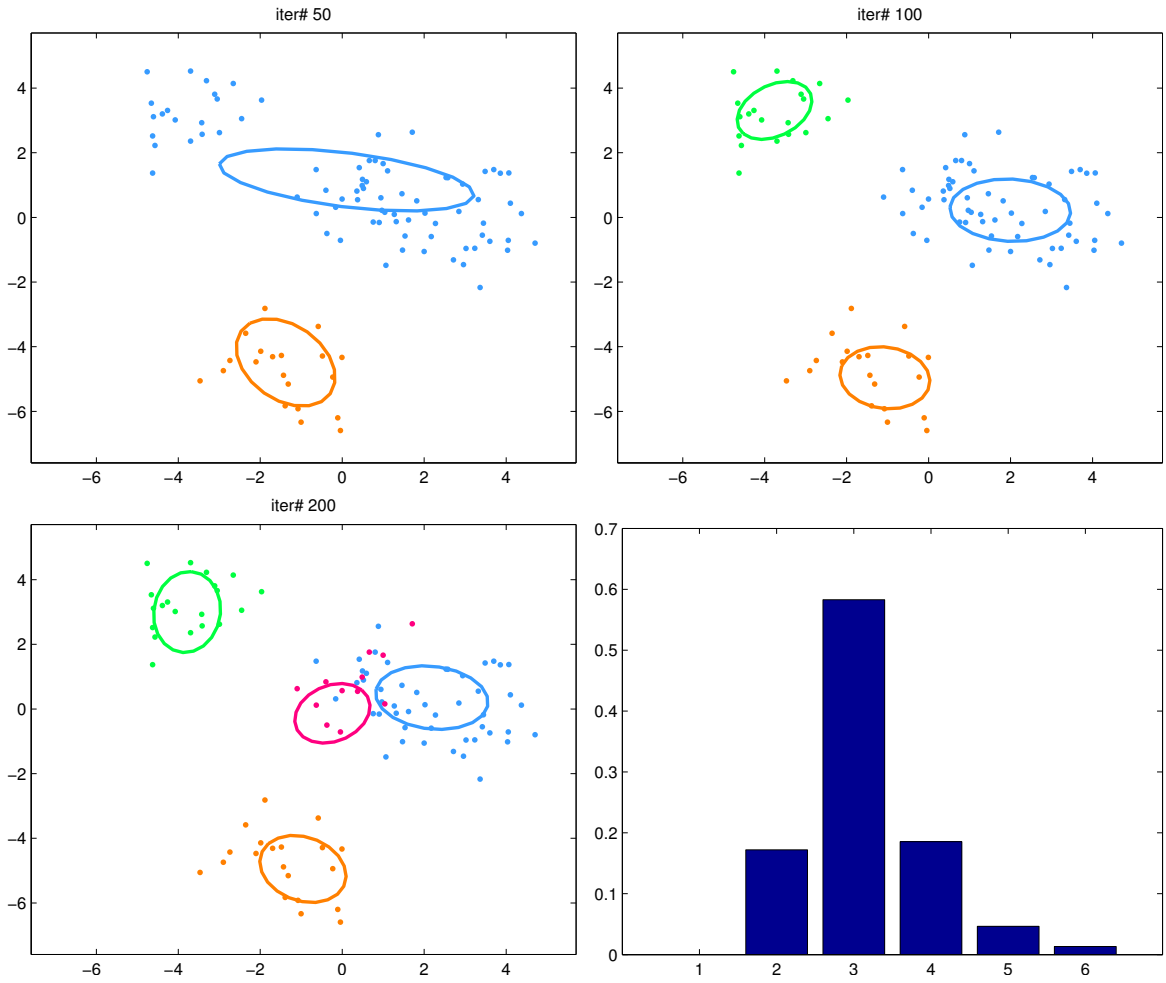


Figure 1.9: (Source: [Mur12])

Lecture 2

Linear Models

Contents

Main Objectives:

- Review the fundamentals of the traditional linear statistical model and related inference procedures.

$\mathcal{O}$

Reading Assignment:

STA511Notes.pdf in BbLearn $\mathcal{R}$

Much of supervised machine learning is founded in linear statistical models, which we quickly review in this chapter, from a frequentist perspective, before moving on to material with more of a machine learning flavor. You should find the material in this chapter to be familiar.

Example 2.1 (Introductory Statistics Model in Matrix Form)**.** *Observation-wise,*

$$y_i \overset{ind}{\sim} N(\mu, \sigma^2) \quad i = 1, \ldots, n.$$

Equivalently, the model may be written

$$y_i = \mu + \epsilon_i \qquad \epsilon_i \overset{ind}{\sim} N(0, \sigma^2) \quad i = 1, \ldots, n.$$

By the way, $\epsilon_i = (y_i - \mu) = (y_i - E(y_i))$, and we see the "error" interpretation.

"Stacking things" and using matrix/vector multiplication/addition, we can write

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \ \boldsymbol{\beta} = [\mu], \ \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}.$$

That is,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} [\mu] + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}.$$

- *Because expectation is a linear operator (i.e., it behaves like finite summation), we have*

$$\begin{aligned} E(\mathbf{y}) &= E(\mathbf{X}\boldsymbol{\beta}+\boldsymbol{\epsilon}) = E(\mathbf{X}\boldsymbol{\beta}) + E(\boldsymbol{\epsilon}) \\ &= \mathbf{X}\boldsymbol{\beta} + \mathbf{0} = \mathbf{X}\boldsymbol{\beta} = \begin{bmatrix} \mu \\ \mu \\ \vdots \\ \mu \end{bmatrix}, \end{aligned}$$

- *The error (vector) variance (matrix) is*

$$\begin{aligned} Var(\boldsymbol{\epsilon}) &= [Cov(\epsilon_i, \epsilon_j)] \\ &= \begin{bmatrix} Var(\epsilon_1) & Cov(\epsilon_1,\epsilon_2) & \cdots & Cov(\epsilon_1,\epsilon_n) \\ Cov(\epsilon_2,\epsilon_1) & Var(\epsilon_2) & \cdots & Cov(\epsilon_2,\epsilon_n) \\ \vdots & & \ddots & \vdots \\ Cov(\epsilon_n,\epsilon_1) & Cov(\epsilon_n,\epsilon_2) & \cdots & Var(\epsilon_n) \end{bmatrix} \\ &= \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix} = \sigma^2 \mathbf{I}_n \end{aligned}$$

- *The observation (vector) variance (matrix) is*

$$\begin{aligned} Var(\mathbf{y}) &= Var(\mathbf{X}\boldsymbol{\beta}+\boldsymbol{\epsilon}) \\ &= Var(\mathbf{X}\boldsymbol{\beta}) + Var(\boldsymbol{\epsilon}) \\ &= 0 + Var(\boldsymbol{\epsilon}) = \sigma^2 \mathbf{I}_n \end{aligned}$$

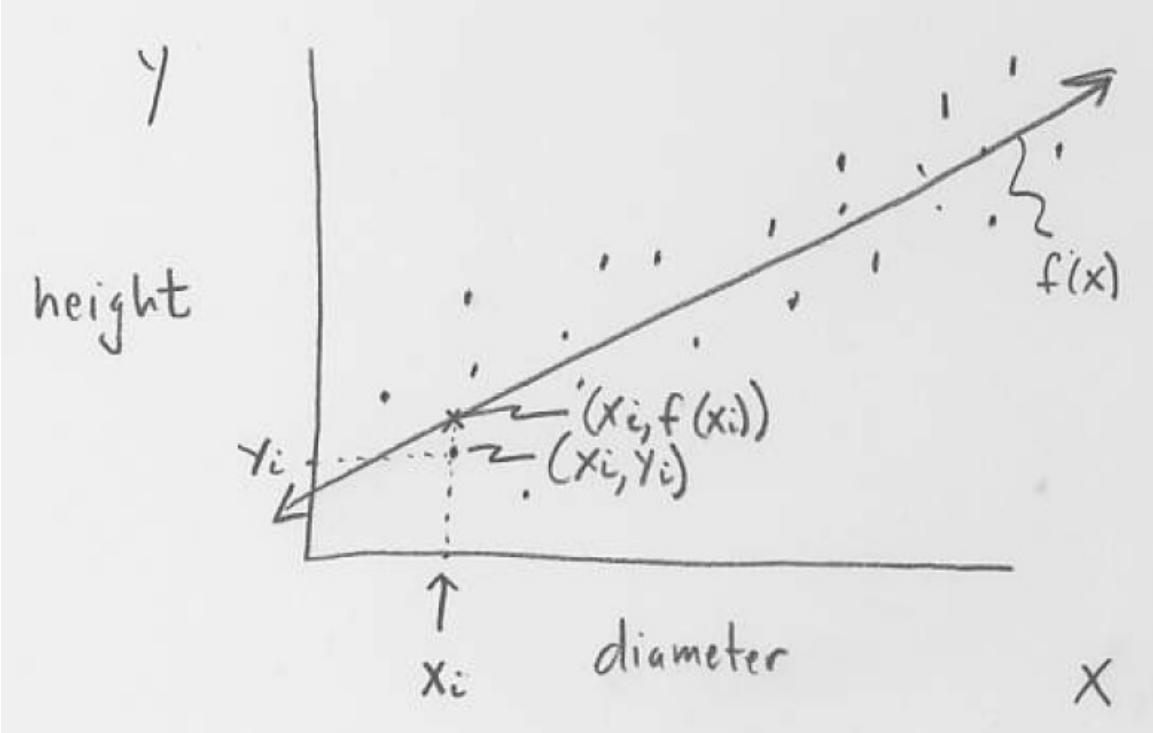
So, we could denote our model as

$$\mathbf{y} \sim N(\mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_n),$$

or as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2 \mathbf{I}_n)$$

Example 2.2 (Simple Linear Regression Model in Matrix Form)**.**
(tree diameter, tree height): (x_i, y_i), $i = 1, \ldots, n$



Observation-wise,

$$\begin{aligned} y_i &= \mu_i + \epsilon_i \quad \textit{where } \mu_i = \mu(x_i, \boldsymbol{\beta}) = \beta_0 + \beta_1 x_i \\ \epsilon_i &\overset{ind}{\sim} N(0, \sigma^2) \end{aligned}$$

or, equivalently,

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \quad \epsilon_i \overset{ind}{\sim} N(0, \sigma^2),$$

or

$$y_i \overset{ind}{\sim} N(\beta_0 + \beta_1 x_i, \sigma^2).$$

As in the previous example, we can write our model as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2 \mathbf{I}_n)$$

where (now a bit differently, but, importantly maintaining the same linear model form),

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_n \end{bmatrix}, \ \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}, \ \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}.$$

That is,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}.$$

And,

$$E(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta} = \begin{bmatrix} \beta_0 + \beta_1 x_1 \\ \beta_0 + \beta_1 x_2 \\ \vdots \\ \beta_0 + \beta_1 x_n \end{bmatrix},$$

$$Var(\boldsymbol{\epsilon}) = \sigma^2 \mathbf{I},$$

$$Var(\mathbf{y}) = \sigma^2 \mathbf{I}.$$

Example 2.3 (Multiple Linear Regression in Matrix Form)**.**
Modeling Assumptions*:*

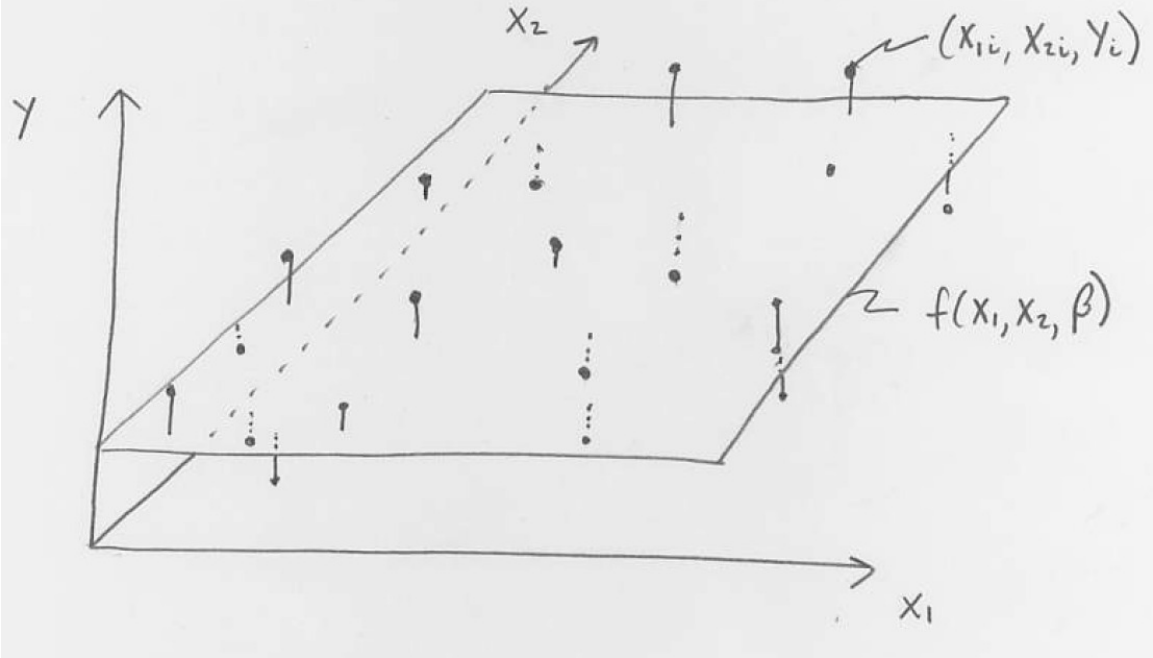
$$y_i = \mu_i + \epsilon_i, \quad \epsilon_i \overset{ind}{\sim} N(0, \sigma^2) \quad i = 1, \ldots, n$$

where

$$\mu_i = \overbrace{\beta_0 + \beta_1 x_{i1} + \cdots + \beta_{p-1} x_{i(p-1)}}^{f(\mathbf{x}_i,\boldsymbol{\beta}) \ or \ \mu(\mathbf{x},\boldsymbol{\beta}) \ or \ \mu(\mathbf{x}_i) \ or \ \mathbf{x}^t\boldsymbol{\beta} \ or \ ...}$$

i.e.,

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_{p-1} x_{i(p-1)} + \epsilon_i.$$



(TYPO: The i subscript of the inputs indicated in the figure should be x_{i1}, x_{i2}, not x_{1i}, x_{2i}.)

You should be able to write the model in ***matrix form****:*

2.1 General Linear Model

- So-called ANOVA (ANalysis Of VAriance) (when inputs are qualitative, or "factor," variables) models can also be written in the same form as illustrated above. See, e.g., [KNNL05].
- In fact, you may be surprised or perhaps disappointed to learn that **a large part of machine learning uses essentially the same model**, perhaps without a formal specification of the full (normal) distribution of errors/observations, depending on whether probability is emphasized or not.
- That is, the **general linear model** encompasses the previous examples, 2.1, 2.2 and 2.3, and a large part of machine learning:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad \boldsymbol{\epsilon} \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma}),$$

  where $\boldsymbol{\Sigma}$ is a general variance (matrix). For much of what we will do, $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$, implying the notion that all outputs are of "**equal quality**" (and independent or, at least, exchangeable).
- However, unlike, let's say, a traditional statistical approach to linear models (e.g., [KNNL05]), and, as emphasized our previous chapter 1 of notes, supervised machine learning emphasizes prediction of unobserved outputs, y, at one or more inputs, $\mathbf{x}$; estimation of and ensuing inference for the parameters of the linear model of the regression function are typically of little or no direct interest.

2.2 Probability Notation

Before moving on with our review a linear models, we pause to introduce notation to represent probability distributions, like the normal (Gaussian) distribution. For example,

$$[\mathbf{y} \,|\, \boldsymbol{\beta}, \boldsymbol{\Sigma}] = \mathrm{N}(\boldsymbol{\mu}(\mathbf{X}, \boldsymbol{\beta}), \boldsymbol{\Sigma}).$$

- Read "|" as "**given**" the quantities on the rhs or "**conditional upon**" the values of the quantities on the rhs being known.
- This notation is short-hand for communicating that the distribution of the random (vector) variable

  $$\mathbf{y} \,|\, \boldsymbol{\beta}, \boldsymbol{\Sigma}$$

  (multiple characters denoting one symbol) (may) depends on (be a function of) $\boldsymbol{\beta}$ and $\boldsymbol{\Sigma}$.
- Often, the quantities on the rhs of | are also specified as random variables, but this is getting ahead of ourselves at the moment.
- The distribution (or random variable) may also depend on other quantities, e.g., $\mathbf{x}$, but explicit conditioning notation is often suppressed when such quantities are not also specified as random variables. For example, traditional regression methods most often assume that inputs (covariates) are known, not requiring a distribution to characterize their uncertainty, and conditioning on inputs is implicit, without notation.
- We will tend to adopt this convention of conditioning notation, though notation varies throughout the scientific literature.
- More later.

2.3 (Ordinary) Least Squares ((O)LS)

How do we get a "good" estimator/estimate of $\mathbf{X}\boldsymbol{\beta}$ or $\boldsymbol{\beta}$?

- As mentioned, supervised **machine learning seeks a "good" predictor** of one or more unobserved ("future") outputs, y. **Generally**

speaking, we want to find a function, $\mu(\mathbf{x})$, that gives us a "good" guess for the unobserved output, y, given input $\mathbf{x}$.

- Generally speaking, if we assume that our observed (and unobserved) data are of "equal quality," then we might think to estimate the function, $\mu(\mathbf{x})$, using the **method of (ordinary) least squares ((O)LS)**, i.e., choose the particular function, denoted $\widehat{\mu}$, that minimizes the objective function (goodness criterion)

$$Q(\mu) = \sum_{i=1}^{n} (y_i - \mu(\mathbf{x}_i))^2.$$

- Unfortunately, **infinitely many functions** satisfy this criterion. We need to somehow **regularize** the problem or to **specify more structure** to the problem to find a unique solution.

- In our current context, e.g., the mean of the **linear model**, $\mu(\mathbf{x}, \boldsymbol{\beta}) = \mathbf{x}^t\boldsymbol{\beta}$, **provides structure** so that we can *typically* find a unique value of $\boldsymbol{\beta}^t = [\beta_0, \ldots, \beta_{p-1}]$, call it $\widehat{\boldsymbol{\beta}}^t = [\widehat{\beta}_0, \ldots, \widehat{\beta}_{p-1}]$, that minimizes

$$Q(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - \mu(\mathbf{x}_i, \boldsymbol{\beta}))^2.$$

- This implies "**equal quality**" of outputs, i.e., each y_i is somehow weighted the same in the LS procedure for finding a good fitting (learning a) value of $\boldsymbol{\beta}$. In our statistical context, this is embodied by the assumption that $\mathrm{Var}(y_i) = \sigma^2$, constant across i, i.e., $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$.

- Incidentally, we might also write Q in **matrix notation**,

$$Q(\boldsymbol{\beta}) = (\mathbf{y} - \boldsymbol{\mu})^t(\mathbf{y} - \boldsymbol{\mu}),$$

- That is, choose the value of $\boldsymbol{\beta}$ such that the distance (squared) between the observation vector $\mathbf{y}$ and its mean vector $\boldsymbol{\mu}$ is minimized.

- Or, more succinctly, **minimize the sum-of-squared errors**,

$$Q(\boldsymbol{\beta}) = \boldsymbol{\epsilon}^t\boldsymbol{\epsilon}.$$

- That is, choose the value of $\boldsymbol{\beta}$ to **minimize the squared length of the error vector**.
- Upon differentiating with respect to the elements of $\boldsymbol{\beta}$ and setting equal to zero (why?), we get the **normal equations**

$$\mathbf{X}^t\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^t\mathbf{y} \qquad \text{details omitted.}$$

  (see, e.g., [KNNL05, p. 201]).
- Almost always in regression—but often not so straight away in ANOVA—we can left multiply by the inverse $(\mathbf{X}^t\mathbf{X})^{-1}$ to get

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{y}.$$

- **Statisticians** typically refer to $\widehat{\boldsymbol{\beta}}$ as the **estimator** or **estimate** of the unknown parameter $\boldsymbol{\beta}$ and to $\widehat{\mu} = \mu(\mathbf{x}, \widehat{\boldsymbol{\beta}})$ as the **fitted model**.
- **Machine learners** might say that we have obtained the learner, $\widehat{\mu} = \mu(\mathbf{x}, \widehat{\boldsymbol{\beta}})$, for predicting new outputs, with learned weight(s) (vector), $\widehat{\boldsymbol{\beta}}$.
- NOTE: The **least squares** criterion for finding estimators usually leads to explicit, **closed-form** solutions for the estimators in the case of ***linear*** models. For non-linear mean models, the least squares criterion typically only implicitly defines estimators, and values of estimator must be found by some iterative algorithm.

2.4 Gauss–Markov Theorem

Aside from being the value of $\boldsymbol{\beta}$ that minimizes sum–of–squared errors, what are other ways in which we might view the "goodness"of the LS estimator $\widehat{\boldsymbol{\beta}}$?

Let

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad \boldsymbol{\epsilon} \sim ?(\mathbf{0}, \sigma^2\mathbf{I}),$$

and consider as above the estimator

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{y}.$$

Then, $\widehat{\boldsymbol{\beta}}$ is called the Best Linear Unbiased Estimator of $\boldsymbol{\beta}$ (BLUE). We don't need to assume $\boldsymbol{\epsilon} \sim \mathrm{N}$ for this result, but we do assume mean zero and variance $\sigma^2\mathbf{I}$, though a very similar version holds for general variance matrix $\boldsymbol{\Sigma}$.

- **Best** in the following sense. Let

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_{p-1} \end{bmatrix}, \ \widehat{\boldsymbol{\beta}} = \begin{bmatrix} \widehat{\beta}_0 \\ \vdots \\ \widehat{\beta}_{p-1} \end{bmatrix},$$

  and let

$$\mathbf{C} = [c_0, \ldots, c_{p-1}]$$

  be any single row matrix of p constants. Then,

$$\mathrm{Var}(\mathbf{C}\widehat{\boldsymbol{\beta}}) \leq \mathrm{Var}(\mathbf{C}\widehat{\boldsymbol{\beta}}^*),$$

  where $\widehat{\boldsymbol{\beta}}^*$ is any estimator you can come up with of the form $\widehat{\boldsymbol{\beta}}^* = \mathbf{a} + \mathbf{A}\mathbf{y}$ with $\mathrm{E}(\widehat{\boldsymbol{\beta}}^*) = \boldsymbol{\beta}$. That is, if you consider any linear unbiased estimator of $\boldsymbol{\beta}$, then linear combinations of $\widehat{\boldsymbol{\beta}}$ have at least as small variances as the same linear combinations of your estimator.

- **Linear** means the estimator is a linear function of the data, i.e, of the form $\mathbf{a} + \mathbf{B}\mathbf{y}$
- **Unbiased** means, as we have seen, $\mathrm{E}(\widehat{\boldsymbol{\beta}}) = \boldsymbol{\beta}$.
- **Estimator** means it is a function of our (random) data vector $\mathbf{y}$ and, in particular, does not depend on $\boldsymbol{\beta}$. (It would be unfortunate if it depended on the unknown quantity being estimated!)

- We'll give some realistic examples of $\boldsymbol{C}$, shortly.

Notes on the Gauss Markov Theorem

- A more general version of GM assumes

$$\boldsymbol{\epsilon} \sim (\mathbf{0}, \boldsymbol{\Sigma})$$

  for general variance matrix $\boldsymbol{\Sigma}$. In this case, the "good" estimator is $\widehat{\boldsymbol{\beta}}_{GLS} = (\mathbf{X}^t\boldsymbol{\Sigma}^{-1}\mathbf{X})^{-1}\mathbf{X}^t\boldsymbol{\Sigma}^{-1}\mathbf{y}$, calld the **generalized least squares (GLS)** estimator of $\boldsymbol{\beta}$.

- In this more general case, $\boldsymbol{\Sigma}$ is **assumed known**, but in practice we almost never know it! It's an unknown parameter (matrix)! If we instead plug in an estimate for it, then this generalized GM theorem is then somehow **out the window**.

- But, as we've seen, special forms of $\boldsymbol{\Sigma}$ allow it to be unknown, and we still get bestness from GM. This includes $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$, which is the form we will use for almost all we do. We still have "bestness"!

- If, in addition to the conditions of the GM, we assume further that $\boldsymbol{\epsilon}$ is normal, then $\widehat{\boldsymbol{\beta}}$ is not only BLUE, it is BUE (Best Unbiased Estimator: best among all functions of the data, linear or not).

- However, BLUE may be far worse than BUE if $\boldsymbol{\epsilon}$ is not normal.

- The goodness of GM depends on our linear statistical model being the **true model** (again, normality is required for BUE but not for BLUE.)

- We will find that, in **machine learning**, we often focus on a search for the **best predictive model**, whether it be **linear, non-linear, explicit or implicit**. We will still use LS (and maximum likelihood and other) methods, but will ultimately use goodness criteria more directly related to prediction (rather than to estimation of $\boldsymbol{\beta}$ like LS and maximum likelihood).
- Thus, our current discussion is somewhat a **tip-of-the-hat to traditional statistics** before we embark on more of a machine learning path.

2.5 Maximum Likelihood

Maximum likelihood (ML) is another method, with its own set of goodness properties, used to estimate μ (and σ^2). We introduce it here, in passing, but will return to it later.

- Assuming normality, $\epsilon \sim N(0, \sigma^2)$, then $y \sim N(\mu(\boldsymbol{x}, \boldsymbol{\beta}), \sigma^2)$, and the (normal) **likelihood function** of n independent observations is

$$L(\boldsymbol{\beta}, \sigma^2) = (2\pi\sigma^2)^{-n/2} \exp\left(-\frac{1}{2} \sum_i \left(\frac{y_i - \mu(\boldsymbol{x}_i, \boldsymbol{\beta})}{\sigma} \right)^2 \right),$$

  and the (estimation) **method of maximum likelihood** tells us to choose the values of the parameters, $\boldsymbol{\beta}$ and σ^2, that maximize the likelihood function, $L(\boldsymbol{\beta}, \sigma^2)$ (with the data fixed at their observed values). That is, **what value of the parameters make the data "most likely"** inasmuchas the likelihood function represents "likeliness?"

- Equivalently, we can find the value of the parameters that maximize the (normal) **log-likelihood function**,

$$l(\boldsymbol{\beta}, \sigma^2) = -\frac{n}{2} \log(2\pi) - n \log(\sigma) - \frac{1}{2} \sum_i \left(\frac{y_i - \mu(\boldsymbol{x}_i, \boldsymbol{\beta})}{\sigma} \right)^2,$$

and we see that **maximizing the (normal) likelihood in $\boldsymbol{\beta}$, for any given $\sigma > 0$, is equivalent to the aforementioned LS criterion of minimizing the sums of squared errors**.

- Again, estimation of σ^2, too, perhaps later.

2.6 STAT 102 Results

- Perhaps feeling good about GM, or about minimizing errors or about maximizing likelihoods, statisticians proceed to conduct statistical inference in a familiar fashion as might be found in an **introductory statistics course**; see **INF511Notes.pdf**.
- In this section, we summarize essential (i.e., relatively programmable?) results from such a course.

- As indicated above, $\widehat{\boldsymbol{\beta}}$ is a linear function of the data vector,

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{y}.$$

- Under the current assumptions of our normal linear model, standard results give the **mean** (vector)

$$\mathrm{E}(\widehat{\boldsymbol{\beta}}) = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathrm{E}(\mathbf{y}) = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{X}\boldsymbol{\beta} = \boldsymbol{\beta}$$

(unbiased for $\boldsymbol{\beta}$),

- **variance** (matrix)

$$\begin{aligned}\mathrm{Var}(\widehat{\boldsymbol{\beta}}) &= (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathrm{Var}(\mathbf{y})\mathbf{X}(\mathbf{X}^t\mathbf{X})^{-1} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\sigma^2\mathbf{I}\mathbf{X}(\mathbf{X}^t\mathbf{X})^{-1} \\ &= \sigma^2(\mathbf{X}^t\mathbf{X})^{-1},\end{aligned}$$

- and **normal distribution**

$$\widehat{\boldsymbol{\beta}} \sim \mathrm{N}(\boldsymbol{\beta}, \sigma^2(\mathbf{X}^t\mathbf{X})^{-1}).$$

- A more general result is that any (non–trivial) **linear combination**(s), $\mathbf{C}\widehat{\boldsymbol{\beta}}$, is also **normal**,

$$\mathbf{C}\widehat{\boldsymbol{\beta}} \sim \mathrm{N}(\mathbf{C}\boldsymbol{\beta}, \sigma^2\mathbf{C}(\mathbf{X}^t\mathbf{X})^{-1}\mathbf{C}^t).$$

  Note that $\mathbf{C}$ will often be a matrix with a single row (a row vector), but $\mathbf{C}$ may also have several rows, each row corresponding to one linear combination, in either a regression or ANOVA context.

- If $\mathbf{C}\boldsymbol{\beta}$ is a **scalar** (i.e., $\mathbf{C}$ is a single row), and we assume **variance is known**, then we can **standardize** in a familiar way to get a $N(0,1)$ rv, and we can perform **z-based inference**. That is,

$$\frac{\mathbf{C}\widehat{\boldsymbol{\beta}} - \mathbf{C}\boldsymbol{\beta}}{\sqrt{\mathrm{Var}(\mathbf{C}\widehat{\boldsymbol{\beta}})}} \sim \mathrm{N}(0,1).$$

- We might also use a $\chi^2(df = 1)$ **distribution** based on what we know about squaring a standard normal. That is,

$$\frac{(\mathbf{C}\widehat{\boldsymbol{\beta}} - \mathbf{C}\boldsymbol{\beta})^2}{\mathrm{Var}(\mathbf{C}\widehat{\boldsymbol{\beta}})} \sim \chi^2(df = 1).$$

- But, we rarely know σ^2, and statisticians will often **estimate (learn?)** σ^2 with

$$\widehat{\sigma}^2 = \frac{\widehat{\boldsymbol{\epsilon}}^t\widehat{\boldsymbol{\epsilon}}}{n-p},$$

  (aka, **MSE**, (estimated) mean-squared-error, mean-squared-residual, etc.) and perform **t-based inference**, when $\mathbf{C}\boldsymbol{\beta}$ is a scalar, using $(n-p)$ error degrees of freedom. That is,

$$\frac{\mathbf{C}\widehat{\boldsymbol{\beta}} - \mathbf{C}\boldsymbol{\beta}}{\sqrt{\widehat{\mathrm{Var}}(\mathbf{C}\widehat{\boldsymbol{\beta}})}} \sim t(n-p).$$

Squaring, we might use an $F(1, n-p)$ distribution. That is,

$$\frac{(\mathbf{C}\widehat{\boldsymbol{\beta}} - \mathbf{C}\boldsymbol{\beta})^2}{\widehat{\mathrm{Var}}(\mathbf{C}\widehat{\boldsymbol{\beta}})} \sim F(1, n-p).$$

(NOTE: $\sqrt{\widehat{\mathrm{Var}}(\mathbf{C}\widehat{\boldsymbol{\beta}})}$ is often called the (estimated) **standard error** of $\mathbf{C}\widehat{\boldsymbol{\beta}}$. Generally, the (estimated) standard deviation of an estimator is called the (estimated) standard error of the estimator.)

- If $\mathbf{C}\boldsymbol{\beta}$ is a **vector** (and assuming again for the moment that **variance is known**), still we can standardize based on what we now about summing the squares of independent standard normals to get a $\chi^2(df = rank(\mathbf{C}))$ random variable. That is,

$$(\mathbf{C}\widehat{\boldsymbol{\beta}} - \mathbf{C}\boldsymbol{\beta})^t(\mathrm{Var}(\mathbf{C}\widehat{\boldsymbol{\beta}}))^{-1}(\mathbf{C}\widehat{\boldsymbol{\beta}} - \mathbf{C}\boldsymbol{\beta}) \sim \chi^2(rank(\mathbf{C}))$$

- **rank** of a matrix is a relatively technical term. Look it up. Usually, in the context of linear model practice, it refers to the number of rows of $\mathbf{C}$, i.e., the number of linear combinations of $\boldsymbol{\beta}$ about which we wish to (simultaneously) infer, usually with a test or interval (region).

- Again, if we **estimate variance**, then we standardize to get an $F(rank(\mathbf{C}), n-p)$, where $n-p$ is the error degrees of freedom (deominator of $MSE = \widehat{\sigma}^2$), p being the number of parameters in the linear mean model.

-

Remark 2.1 (F Statistic)**.** *In short, for general* $\mathbf{C}$ *(full row rank), and estimating* σ^2 *with* $\widehat{\sigma}^2 = MSE$*, we have*

$$(\mathbf{C}\widehat{\boldsymbol{\beta}} - \mathbf{C}\boldsymbol{\beta})^t(\widehat{Var}(\mathbf{C}\widehat{\boldsymbol{\beta}}))^{-1}(\mathbf{C}\widehat{\boldsymbol{\beta}} - \mathbf{C}\boldsymbol{\beta}) \sim F(rank(\mathbf{C}), n-p).$$

That is,

$$(\mathbf{C}\widehat{\boldsymbol{\beta}} - \mathbf{C}\boldsymbol{\beta})^t(\widehat{\sigma}^2\mathbf{C}(\mathbf{X}^t\mathbf{X})^{-1}\mathbf{C}^t)^{-1}(\mathbf{C}\widehat{\boldsymbol{\beta}} - \mathbf{C}\boldsymbol{\beta}) \sim F(rank(\mathbf{C}), n-p).$$

This result is used a lot in both regression and ANOVA.

Thus, given the assumptions of the General Linear Model, we have several distributions, related to $\widehat{\boldsymbol{\beta}}$, to help us conduct inference, i.e., to compute test p-values and obtain confidence intervals with some level, as in an introductory statistics course. Usually, as you know, we work with t and F, though z and χ^2 may sometime be used. Below, we give examples to illustrate these fundamental results.

Much of the material presented here will serve us later when we cover Bayesian linear models. Not only will the same quantities appear (i.e., same matrices and vectors), but also the same inferences, in some sense.

2.7 Examples of STAT 102 Results Using R

We use a small, simple data set to illustrate much of the results just presented, above. (Again, this is a sort of **tip of the hat to traditional statistics** as we begin to move toward a more ML perspective.) The data are measurements of the distance between Earth and $n = 24$ nebulae (y) and the velocity with which these nebulae are traveling from Earth (x or $\mathbf{x}^t = (1, \text{velocity})$). I call these data the **big bang data**. See [RS13, Ch. 7] and the related `R` library package `Sleuth3` for more information.

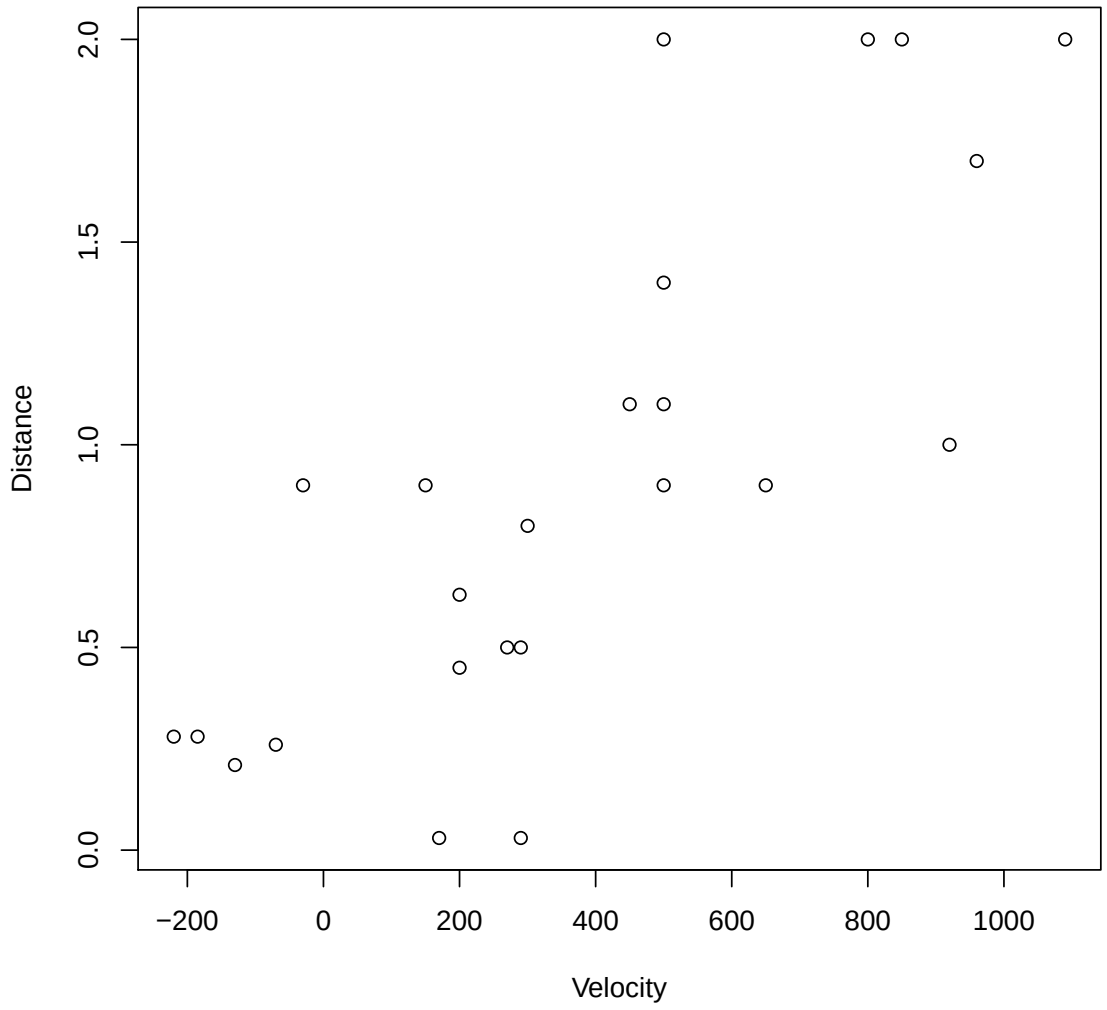
- The next chunk simply reads the data into a data frame and gives a quick look at it—check for gross errors.

```
> ## LS estimation of beta
> ## File is called case0701 in Sleuth3 package
> library(Sleuth3)
> bigbang.df<- case0701
> detach(package:Sleuth3)
> ## Or, more succintly, dbigbang.df<- Sleuth3::case0701
> names(bigbang.df)

[1] "Velocity" "Distance"

> head(bigbang.df)

  Velocity Distance
1      170     0.03
```

```
2      290     0.03
3     -130     0.21
4      -70     0.26
5     -185     0.28
6     -220     0.28
```

- Quick graphical EDA—check for gross errors.

```
> plot(Distance ~ Velocity, data=bigbang.df)
```



- Prepare model components mentioned in above typeset notes.

```
> y<- bigbang.df$Distance
> X<- model.matrix(Distance ~ Velocity, data=bigbang.df)
> head(y,n=3); tail(y,n=3)

[1] 0.03 0.03 0.21
[1] 2 2 2

> (n<-dim(X)[1]); (p<- dim(X)[2]); head(X,n=3); tail(X,n=3)

[1] 24
[1] 2
  (Intercept) Velocity
1           1      170
2           1      290
3           1     -130
   (Intercept) Velocity
22           1      850
23           1      800
24           1     1090
```

- Obtain quantities discussed in above typeset notes.

```
> (betahat<- (XtXinv<- solve( XtX<-t(X)%*%X ))%*%t(X)%*%y)

                   [,1]
(Intercept) 0.399170440
Velocity    0.001372408

> ehat<- (y - X%*%betahat)
> ## MSE (estimator of sigma^2)
> (sig2hat<- as.vector(t(ehat) %*% ehat / (n - p)))

[1] 0.1645359

> sqrt(sig2hat)

[1] 0.4056302
```

```
> varBhat<- sig2hat * XtXinv
> (seBhat<- sqrt(diag(varBhat)))

 (Intercept)     Velocity
0.1186661507 0.0002278214

> ## More automatically
> summary(bigbang.lm<- lm(Distance ~ Velocity, data=bigbang.df))


Call:
lm(formula = Distance ~ Velocity, data = bigbang.df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.76717 -0.23517 -0.01083  0.21081  0.91463

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.3991704  0.1186662   3.364   0.0028
Velocity    0.0013724  0.0002278   6.024 4.61e-06

(Intercept) **
Velocity    ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4056 on 22 degrees of freedom
Multiple R-squared:  0.6226,Adjusted R-squared:  0.6054
F-statistic: 36.29 on 1 and 22 DF,  p-value: 4.608e-06
```

- t-based confidence interval estimation of **slope** using the general linear hypothesis/combination, i.e., "$\mathbf{C}\boldsymbol{\beta}$" approach, as discussed in above typeset notes.

```
> ## slope interval
> (Cmat<- matrix(c(0,1),nrow=1,ncol=p))

     [,1] [,2]
[1,]    0    1
```

```
> (CBhat<- as.vector(Cmat%*%betahat))

[1] 0.001372408

> (CBse<- as.vector(sqrt(Cmat %*% varBhat %*% t(Cmat))))

[1] 0.0002278214

> (tmult<- qt(1-0.05/2, df=n-p))

[1] 2.073873

> CBhat + c(-1,1) * CBse * tmult

[1] 0.0008999349 0.0018448801
```

```
> ## slope interval more automatically
> confint(bigbang.lm, parm=2)

                2.5 %     97.5 %
Velocity 0.0008999349 0.00184488
```

- t-based test of **intercept** using the general linear hypothesis/combination, i.e., "$\mathbf{C}\boldsymbol{\beta}$" approach, as discussed in above typeset notes.

```
> ## t-test intercept
> (Cmat<- matrix(c(1,0),nrow=1,ncol=p))

     [,1] [,2]
[1,]    1    0

> (CBhat<- as.vector(Cmat%*%betahat))

[1] 0.3991704

> (CBse<- as.vector(sqrt(CBvar<- Cmat %*% varBhat %*% t(Cmat))))

[1] 0.1186662
```

```
> CB0<- 0
> (tstat<- (CBhat - CB0) / CBse)

[1] 3.36381

> (pval<- 2 * pt(abs(tstat), df=n-p, lower=FALSE))

[1] 0.002803008

> (reject <- pval <= 0.05)

[1] TRUE

> ## equivalent (if unconventional) F-test
> (Fstat<- as.vector(t(CBhat - CB0) %*% solve(CBvar) %*%
+       (Cmat%*%betahat - CB0)))

[1] 11.31522

> (pval<- pf(Fstat, 1, n-p, lower.tail=FALSE))

[1] 0.002803008

> tstat^2 == Fstat

[1] TRUE

> ## test intercept more automatically: see summary results, above
```

- t-based confidence interval estimation of **mean** using the general linear hypothesis/combination, i.e., "$\mathbf{C}\boldsymbol{\beta}$" approach, as discussed in above typeset notes.

```
> ## Confidence interval for mean, E(y) = mu(x,beta), at x = (1, 800)^t
> (Cmat<- matrix(c(1,800),nrow=1,ncol=p))

     [,1] [,2]
[1,]    1  800
```

```
> (CBhat<- as.vector(Cmat%*%betahat))

[1] 1.497096

> (CBse<- as.vector(sqrt(Cmat %*% varBhat %*% t(Cmat))))

[1] 0.1277242

> (tmult<- qt(1-0.05/2, df=n-p))

[1] 2.073873

> CBhat + c(-1,1) * CBse * tmult

[1] 1.232213 1.761980

> ## Confidence interval more automatically
> xnew<- data.frame(Velocity=800)
> predict(bigbang.lm, newdata=xnew, se.fit=TRUE, interval="confidence",
+         level=0.95)

$fit
       fit      lwr     upr
1 1.497096 1.232213 1.76198

$se.fit
[1] 0.1277242

$df
[1] 22

$residual.scale
[1] 0.4056302
```

- t-based prediction interval estimation of y using the general linear hypothesis/combination, i.e., "$\mathbf{C}\boldsymbol{\beta}$" approach, as discussed in above typeset notes.

```
> ## Prediction interval for new y = mu(x,beta) + eps at x = (1, 800)^t
> (Cmat<- matrix(c(1,800),nrow=1,ncol=p))

     [,1] [,2]
[1,]    1  800

> (CBhat<- as.vector(Cmat%*%betahat))

[1] 1.497096

> (CBse<- as.vector(sqrt(sig2hat + Cmat %*% varBhat %*% t(Cmat))))

[1] 0.4252638

> (tmult<- qt(1-0.05/2, df=n-p))

[1] 2.073873

> CBhat + c(-1,1) * CBse * tmult

[1] 0.6151532 2.3790397

> ## Prediction interval more automatically
> predict(bigbang.lm, newdata=xnew, se.fit=TRUE, interval="predict",
+         level=0.95)

$fit
       fit       lwr     upr
1 1.497096 0.6151532 2.37904

$se.fit
[1] 0.1277242

$df
[1] 22

$residual.scale
[1] 0.4056302
```

- We may further illustrate the F-based result 2.1, later, when we introduce **model/variable selection**.
- BTW, the `glh.test` and `estimable` functions in the `R` library package, `gmodels`, are more generally useful functions for this sort of "$\mathbf{C}\boldsymbol{\beta}$" inference approach.

Lecture 3

Variable Selection for Linear Models

Contents

Main Objectives:

- Other goodness criteria besides LS or ML (Max. Like.):
  - AIC
  - BIC
  - C_P
  - adjusted R^2
  - generalization error (aka test error or prediction error)
- An exhaustive search through all possible subsets of inputs using a best subsets procedure.
- Note other, non-exhaustive procedures: forward stepwise selection (for a large number of inputs) or backwards stepwise selection
- Validation set procedure and estimated generalization error
- Criteria as adjustments of training MSE toward generalization error
- K-fold cross-validation (CV) toward generalization error
- `regsubsets` in the `leaps` package for an implementation of a best subsets procedure.
- Realize, again, that we are deliberately beginning with a relatively classical statistical perspective/methods and working our way toward more of a ML (or, at least, statistical learning) perspective. Hopefully, we do this in a seamless manner that leaves you relatively agnostic about what banner the methods fall under. (Or, somehow, this allows you to use polite jargon when speaking with devotees of one or the "other." (You my find "statistical" devotees to be a bit older!)) Also, you may get some sense for the historical development of machine learning (from a statistical perspective).

$\mathcal{O}$

Reading Assignment:

Recommended: [JWHT14, Ch. 5 & 6] and [KNNL05, §9.4]

Others: [HTF01, §3.3.1], [Wak13, §4.8.2], INF511Notes.pdf $\mathcal{R}$

3.1 GLH ($C\beta$) Approach To (Input) Variable Selection

- As you will see in the notes for INF 511 (BbLearn pdf file INF511Notes.pdf), as summarized in our previous lecture chapter 2, we can use the "$\mathbf{C\beta}$" approach (aka, GLH approach or **full vs. reduced model approach** or **extra-sum-of-squares approach**...).
- But, that approach, unmodified, at least, seems unsuitable or somehow crude for exploring subsets of input variables for even a moderate number of inputs.
- In any case, this approach is **not terribly popular** among machine learners, and we will not emphasize it except to say that it may show up briefly in an assignment. (This is not to say that it isn't useful for some circumstances as illustrated previously, if only for comparison to other methods.)

3.2 Selection Criteria

Before getting to procedures for searching for "good" subsets of variables to learn/fit a (linear) model, we introduce selection criteria, beyond LS (least squares) or ML (maximum likelihood, not machine learning), which serve us in selecting among models/subsets of inputs that our search procedures provide to us. Some of these criteria are applicable beyond the (normal) linear model. We will learn why we may not want to use all inputs (usually not without somehow attending to the pitfalls of such an approach).

3.2.1 Adjusted R^2

The ("unadjusted") **coefficient of determination** is

$$\begin{aligned} R^2 &= 1 - \frac{SSE}{SSTO}, \quad \text{where} \\ SSE &= (\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})^t(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}) \\ &= \widehat{\boldsymbol{\epsilon}}^t\widehat{\boldsymbol{\epsilon}}, \quad \text{and} \\ SSTO &= (\mathbf{y} - \mathbf{1}\bar{y})^t(\mathbf{y} - \mathbf{1}\bar{y}). \end{aligned}$$

- Loosely speaking, a higher R^2 is better, though this is simplistic without the context of a more thorough analysis.
- Because we can always drive R^2 to its maximum of 1 by increasing the complexity (number of inputs), giving a "perfect" fit to the n data points, R^2 is not such a popular model/subset selection criterion on its own. (Without addressing the potential pitfalls (later) of including all inputs, such a learner typically has poor predictive ability and essentially no ability to reduce data to allow for meaningful interpretation, though machine learners typically do not care about interpretation beyond prediction.)
- R^2 and SSE will produce equivalent model/subset selections.

- **Adjusted** R^2 adjusts R^2 for model complexity, penalizing models that are somehow too complex in a sense to be made clear as we go.
- The "best" model/subset is the one with the highest adjusted R^2, defined as

$$\begin{aligned} R^2_a &= 1 - \frac{SSE/(n-p)}{SSTO/(n-1)}, \\ &= 1 - (1 - R^2)\left(\frac{n-1}{n-p}\right) \end{aligned}$$

- The rough idea is that you **spend** p **degrees of freedom**, which would otherwise be used to estimate, e.g., tighter confidence intervals, in the **hope of buying a smaller** SSE to justify the expense with a sufficiently smaller error, σ^2, as estimated by $\widehat{\sigma}^2 = MSE = SSE/(n-p)$, hence a larger adjusted R^2; the expense is no longer justified when you get relatively little in return for spending additional degrees of freedom, as indicated by a decreasing adjusted R^2 with additional inputs.
- (What's the typical "STAT 101" expression of a confidence interval?)

3.2.2 C_P

Mallow's C_P ("C" in honor of"C"uthbert Daniels, *not* for Colin Mallow's as Wikipedia would have you believe), applicable to normal linear models, is (almost) equivalent to the more generally applicable, AIC, when AIC is restricted to normal linear models; see AIC below.

- Let P denote a (possibly empty) subset of k potential inputs that we may index by values in $\{1, \ldots, k\}$.
- Let $|P|$ $(0 \leq |P| \leq k)$ denote the number of inputs in input subset P.
- Then, there are $p = (|P| + 1)$ parameters (weights) in the model defined by the input subset, P. (Potentially confusing, I know.)
- E.g., $p = (|P| + 1) = (k + 1)$ for the model having all k inputs, the full model, so-to-speak.

$$\begin{aligned} C_P &= \frac{SSE_P}{\widehat{\sigma}^2} - (n - 2p) \quad \text{where} \\ SSE_P &= (\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}_P)^t(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}_P), \end{aligned}$$

$\widehat{\boldsymbol{\beta}}_P$ is the LS estimate of the parameter (weight), $\boldsymbol{\beta}_P$, for model with subset P, and $\widehat{\sigma}^2 = SSE_k/(n-k-1)$, i.e., the MSE for the fullest model, with all k inputs.

- Choose the model/subset with small values of $C_P \approx p = |P| + 1$ (number of parameters). (Explanatory/interpretive details omitted.)
- Assumes that the fullest model estimate $\widehat{\sigma}^2$ is a good estimate of σ^2 (i.e., that the full model is unbiased giving $\widehat{\sigma}^2$ that is an unbiased estimator for σ^2).

3.2.3 *AIC*

AIC, Akaike's An Information Criterion (aka, usually, **Akaike's Information Criterion**) is a **penalized likelihood** statistic, and is a generalization of C_P

$$AIC \;=\; -2l(\widehat{\boldsymbol{\theta}}_P) + 2p$$

where $l(\widehat{\boldsymbol{\theta}}_P)$ denotes the maximum of the log-likelihood function (evaluated at the denoted MLE $\widehat{\boldsymbol{\theta}}_P$) for model P having p parameters. We generically denote the parameter as $\boldsymbol{\theta}_P$ here because AIC applies more generally than to linear models, where, in this case, $\boldsymbol{\theta}_P = \boldsymbol{\beta}_P$ (and we assume σ^2 is known).

- Choose the model/subset with **lowest** AIC.
- For normal linear statistical models, AIC and C_P will produce equivalent model/subset selections, assuming $\sigma^2 = \widehat{\sigma}^2$ is known.

3.2.4 BIC

BIC (Bayesian Information Criterion, aka, Schwarz's Bayesian Criterion (SBC)) is another penalized likelihood statistic,

$$BIC \;=\; -2l(\widehat{\boldsymbol{\theta}}_P) + p\log(n)$$

- Choose the model/subset with **lowest** BIC.
- Note that log denotes natural log, and $\log(n) > 2$ when $n > \exp(2) \approx 7$.
- Thus, BIC penalizes large (more complex models) more heavily than (i.e., tends to suggest simpler models compared to) AIC for most realistic sample sizes, n.

3.3 Best Subsets

To introduce the best subsets procedure, consider the example where we have $k = 4$ (not a large number of) inputs, and we label inputs generically as x_1, x_2, x_3, x_4. (We do not often consider whether the "input" 1, with an intercept parameter, should be in the model or not.) Then there are $2^4 = 16$ possible models reflecting all possible subsets of the $k = 4$ inputs:

- All four **1-input** models: x_1; x_2; x_3; x_4
- All six **2-input** models: x_1,x_2; x_1,x_3; x_1,x_4; x_2,x_3; x_2,x_4; x_3,x_4.
- All four **3-input** models: x_1,x_2,x_3; x_1,x_2,x_4; x_1,x_3,x_4; x_2,x_3,x_4.
- One **4-input** model: x_1,x_2,x_3,x_4.
- One **0-input** model: (just $x_0 = 1$ for the intercept parameter; we do not often omit the "input" 1).
- Generally, if there are k potential inputs (not including 1) to choose from, then there are 2^k possible regression models (subsets of inputs) to be considered, somehow.

- For example, $2^{10} = 1,024$ models (kilobyte); $2^{20} = 1,048,576$ (megabyte), $2^{30} = 1,073,741,824$ (gigabyte), and $2^{40} = 1,099,511,627,776$ (terabyte).
- A **best subsets** procedure uses one or more criteria (see above, and generalization error, below) to select the best model/subset from all possible models/subsets.
- Because it **searches through all possible models**, it is called an **exhaustive** procedure
- Clearly, this will break down computationally with a moderate number k of inputs.

3.4 Examples in R

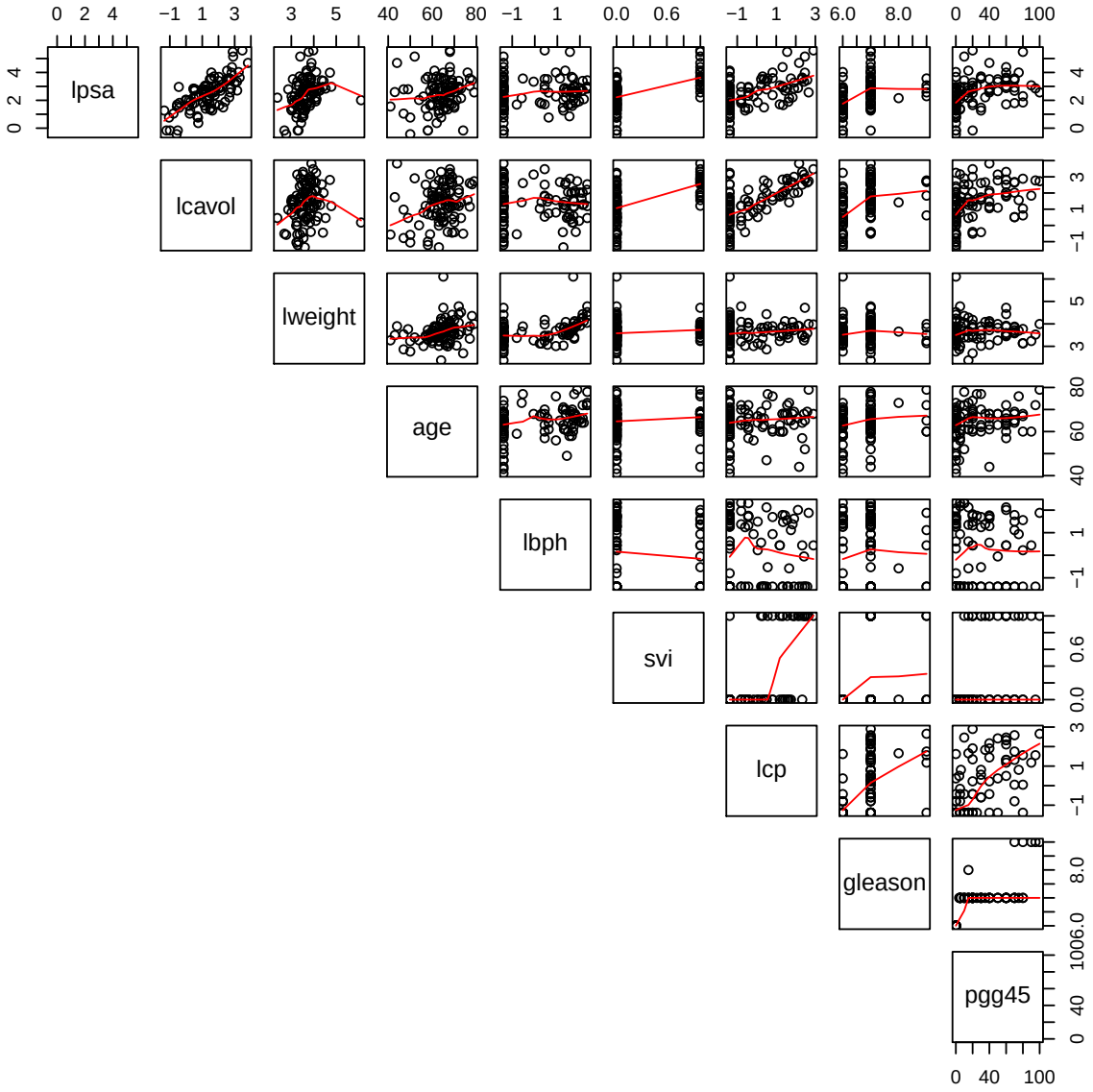
- We use the `regsubsets` function, in the `R` library package `leaps`, on the `Prostate` data set in the `lasso2` package.
- You will find more information via `help(Prostate)` (after submitting `library(lasso2)`).
- If `lpsa` is the **output**, y, what are some of the potential values for **inputs**? (See code below.)
- ...for P, $|P|$, p, k...?

```
> library(lasso2)

R Package to solve regression problems while imposing
an L1 constraint on the parameters.  Based on S-plus Release 2.1
Copyright (C) 1998, 1999
Justin Lokhorst  <jlokhors@stats.adelaide.edu.au>
```

```
Berwin A. Turlach <bturlach@stats.adelaide.edu.au>
Bill Venables    <wvenable@stats.adelaide.edu.au>
   Copyright (C) 2002
Martin Maechler <maechler@stat.math.ethz.ch>
```

```
> data(Prostate)
> dim(Prostate)

[1] 97  9

> names(Prostate)

[1] "lcavol"  "lweight" "age"     "lbph"
[5] "svi"     "lcp"     "gleason" "pgg45"
[9] "lpsa"
```

```
> pairs(lpsa ~ . ,
+       upper.panel = panel.smooth,
+       lower.panel = NULL,
+       data= Prostate)
```



The following are some of the available options for the `regsubsets` function.

- `nbest`= the number of best subsets at each model (subset) size
- `nvmax`= maximum number of variables (model/subset size) to consider

- `force.in`= names or indices of variable to always have in model
- `force.out`= names or indices of variable to never have in model
- `method`=c('exhaustive', 'backward', 'forward', 'seqrep')

- The **next chunk** gives a summary of the best subsets procedure as implemented in the `regsubsets` function.
- It reports the `nbest=6` best models *for each size subset* according to SSE (which it calls rss), up to maximum subset size `nvmax=8`.
- (Note that, *for models of the same size*, our criteria are equivalent to SSE, but each criterion does penalize in its own way across size. In other words, for models of each size, we can use SSE to select the best (of a size), then use our favorite criteria to select the best model across size. Thus, we could use SSE to reduce $2^8 - 1 = 255$ models to just 8 models, then compare these 8 using our favorite criteria (other than SSE); we don't consider the (one) model with no covariates here.)

```
> library(leaps)
> wold<- getOption("width")
> options(width=75)
> Prostate.rsub<- regsubsets(lpsa ~ ., data=Prostate,
+                            nbest=6,
+                            nvmax=8)
> (prsub.sum<- summary(Prostate.rsub))

Subset selection object
Call: regsubsets.formula(lpsa ~ ., data = Prostate, nbest = 6, nvmax = 8)
8 Variables  (and intercept)
        Forced in Forced out
lcavol      FALSE      FALSE
```

```
lweight     FALSE      FALSE
age         FALSE      FALSE
lbph        FALSE      FALSE
svi         FALSE      FALSE
lcp         FALSE      FALSE
gleason     FALSE      FALSE
pgg45       FALSE      FALSE
6 subsets of each size up to 8
Selection Algorithm: exhaustive
         lcavol lweight age lbph svi lcp gleason pgg45
1  ( 1 ) "*"    " "     " " " "  " " " " " "     " "
1  ( 2 ) " "    " "     " " " "  "*" " " " "     " "
1  ( 3 ) " "    " "     " " " "  " " "*" " "     " "
1  ( 4 ) " "    " "     " " " "  " " " " " "     "*"
1  ( 5 ) " "    " "     " " " "  " " " " "*"     " "
1  ( 6 ) " "    "*"     " " " "  " " " " " "     " "
2  ( 1 ) "*"    "*"     " " " "  " " " " " "     " "
2  ( 2 ) "*"    " "     " " " "  "*" " " " "     " "
2  ( 3 ) "*"    " "     " " "*"  " " " " " "     " "
2  ( 4 ) "*"    " "     " " " "  " " " " " "     "*"
2  ( 5 ) "*"    " "     " " " "  " " "*" " "     " "
2  ( 6 ) "*"    " "     " " " "  " " " " "*"     " "
3  ( 1 ) "*"    "*"     " " " "  "*" " " " "     " "
3  ( 2 ) "*"    " "     " " "*"  "*" " " " "     " "
3  ( 3 ) "*"    "*"     " " " "  " " " " " "     "*"
3  ( 4 ) "*"    "*"     " " " "  " " "*" " "     " "
3  ( 5 ) "*"    "*"     " " " "  " " " " "*"     " "
3  ( 6 ) "*"    "*"     " " "*"  " " " " " "     " "
4  ( 1 ) "*"    "*"     " " "*"  "*" " " " "     " "
4  ( 2 ) "*"    "*"     " " " "  "*" " " " "     "*"
4  ( 3 ) "*"    "*"     "*" " "  "*" " " " "     " "
4  ( 4 ) "*"    "*"     " " " "  "*" " " "*"     " "
4  ( 5 ) "*"    "*"     " " " "  "*" "*" " "     " "
4  ( 6 ) "*"    " "     "*" "*"  "*" " " " "     " "
5  ( 1 ) "*"    "*"     "*" "*"  "*" " " " "     " "
5  ( 2 ) "*"    "*"     " " "*"  "*" " " " "     "*"
5  ( 3 ) "*"    "*"     " " "*"  "*" " " "*"     " "
5  ( 4 ) "*"    "*"     " " "*"  "*" "*" " "     " "
5  ( 5 ) "*"    "*"     "*" " "  "*" " " " "     "*"
5  ( 6 ) "*"    "*"     "*" " "  "*" " " "*"     " "
6  ( 1 ) "*"    "*"     "*" "*"  "*" " " " "     "*"
6  ( 2 ) "*"    "*"     "*" "*"  "*" " " "*"     " "
```

```
6  ( 3 ) "*"    "*"     "*" "*"  "*" "*" " "     " "
6  ( 4 ) "*"    "*"     " " "*"  "*" "*" " "     "*"
6  ( 5 ) "*"    "*"     "*" " "  "*" "*" " "     "*"
6  ( 6 ) "*"    "*"     " " "*"  "*" "*" "*"     " "
7  ( 1 ) "*"    "*"     "*" "*"  "*" "*" " "     "*"
7  ( 2 ) "*"    "*"     "*" "*"  "*" "*" "*"     " "
7  ( 3 ) "*"    "*"     "*" "*"  "*" " " "*"     "*"
7  ( 4 ) "*"    "*"     " " "*"  "*" "*" "*"     "*"
7  ( 5 ) "*"    "*"     "*" " "  "*" "*" "*"     "*"
7  ( 6 ) "*"    " "     "*" "*"  "*" "*" "*"     "*"
8  ( 1 ) "*"    "*"     "*" "*"  "*" "*" "*"     "*"

> options(width=wold)
```

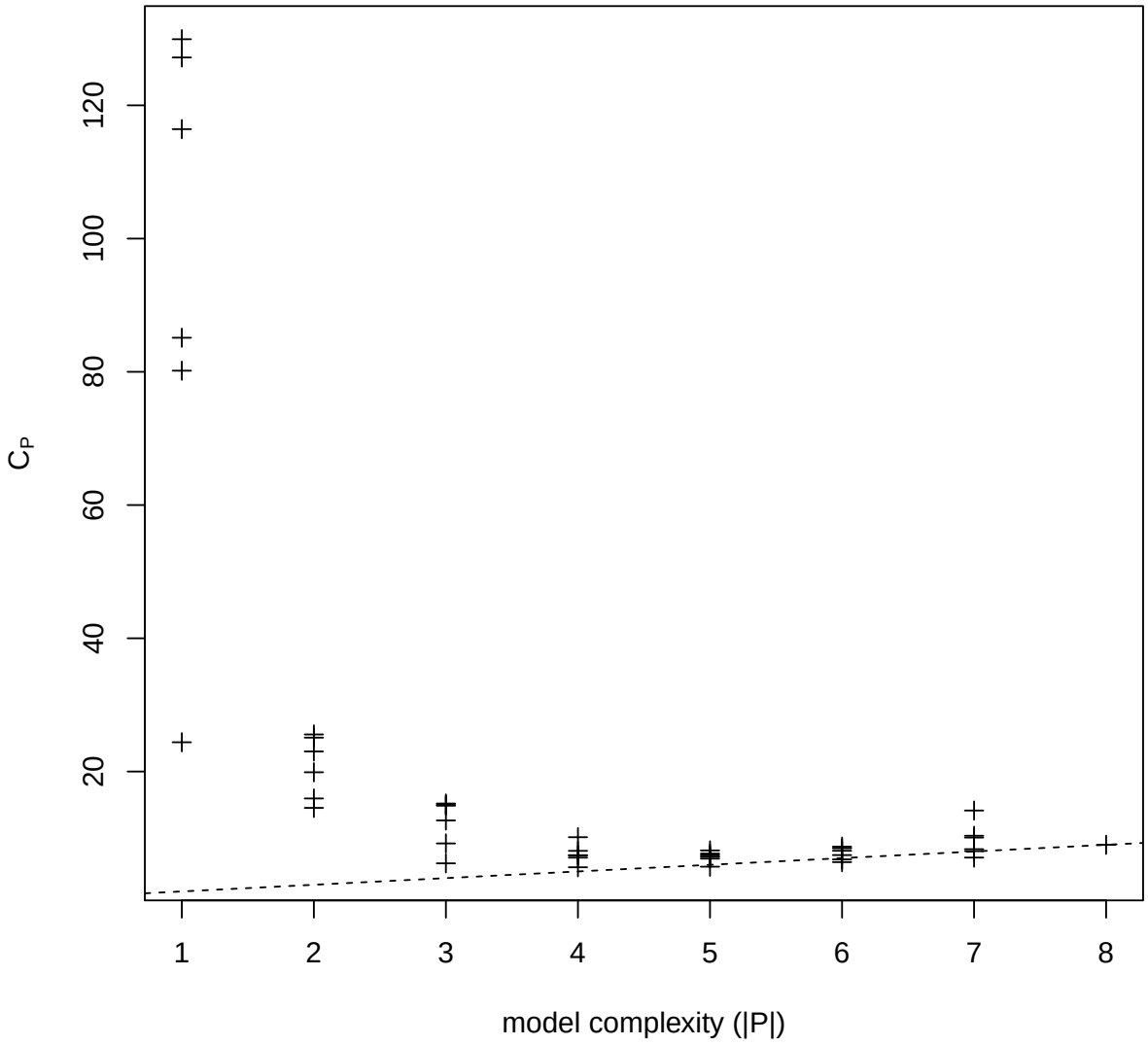
Also, we can choose a best model, over all sizes, according to one of our criteria, which **regsubsets** computes for us, as indicated in the next chunk.

```
> names(prsub.sum)

[1] "which"  "rsq"    "rss"    "adjr2"  "cp"
[6] "bic"    "outmat" "obj"

> attach(prsub.sum)
> pstate.cri<- cbind.data.frame(
+     P=as.numeric(substr(rownames(prsub.sum$outmat), 1,2)),
+     rsq=rsq, rss=rss, adjr2=adjr2, cp=cp, bic=bic)
> detach(prsub.sum)
> names(pstate.cri)

[1] "P"     "rsq"   "rss"   "adjr2" "cp"
[6] "bic"
```

- For example, the next chunk plots C_P vs. **model complexity**, i.e., vs. the number of inputs ($|P|$), i.e., vs. subset size.

- As mentioned above, the C_P criterion tells us to select models (input subsets) with small C_P values near $p = (|P| + 1)$.
- BTW, the `subsets` function in the `car` package offers additional plot functionality.

```
> plot(cp ~ P, data=pstate.cri, pch=3,
+      xlab="model complexity (|P|)",
+      ylab=expression(C[P]))
> ## Reference line where CP = p (= 1 + |P| = 1 + k)
> abline(c(1,1), lty=2)
```

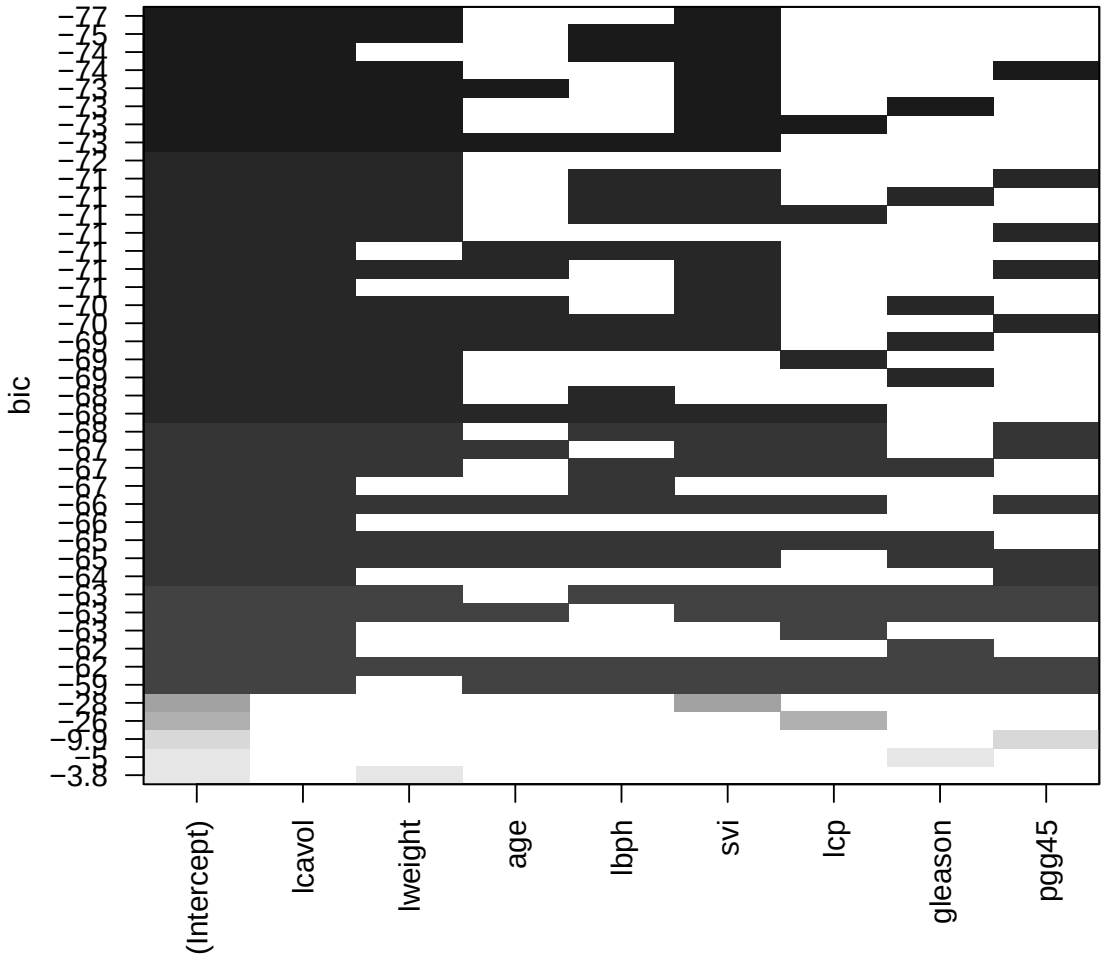



```
> ## Best in terms of CP: choose minimum (like AIC) and
> ## (hopefully) CP approx p=|P|+1 = k+1 (unlike AIC)
> prsub.sum$outmat[which.min(prsub.sum$cp),]

 lcavol lweight     age    lbph     svi     lcp
    "*"     "*"     " "     "*"     "*"     " "
gleason   pgg45
    " "     " "

> prsub.sum$cp[which.min(prsub.sum$cp)] ## approx p=4+1=5
```

```
[1] 5.626398
```

Perhaps you prefer the plot functionality built into the `leaps` package, as illustrated in the next chunk, using BIC.

```
> plot(Prostate.rsub, scale="bic",
+      col=gray(seq(0.1,0.9,length=16)))
```



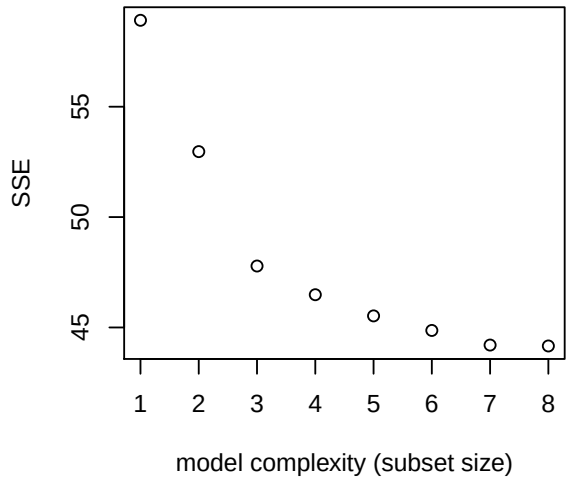
```
> ## car::subsets(Prostate.rsub)  ## not run
```

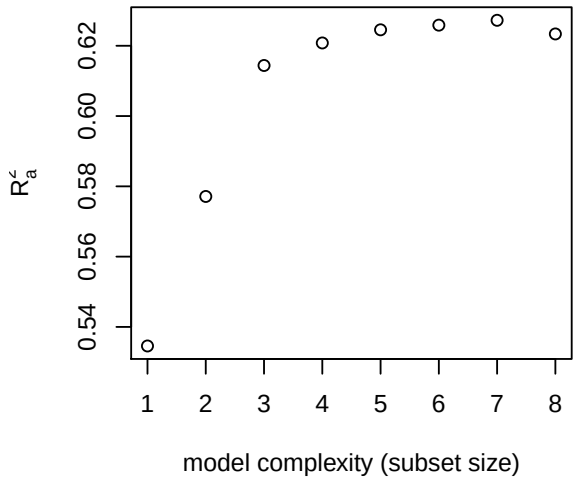
```
> ## Best in terms of BIC
> prsub.sum$outmat[which.min(prsub.sum$bic),]

 lcavol lweight     age    lbph     svi     lcp
    "*"     "*"     " "     " "     "*"     " "
gleason   pgg45
    " "     " "

> coef(prsub.sum$obj, which.min(prsub.sum$bic))

(Intercept)      lcavol     lweight         svi
 -0.2680724   0.5516386   0.5085359   0.6661583

> ## Perhaps get familiar lm fitted object (not run)
> ## coef(bestbicmod<- lm(lpsa ~ lcavol+lweight+svi,
> ##                      data=Prostate))
> ## Perhaps predict using predict function...(not shown)
```

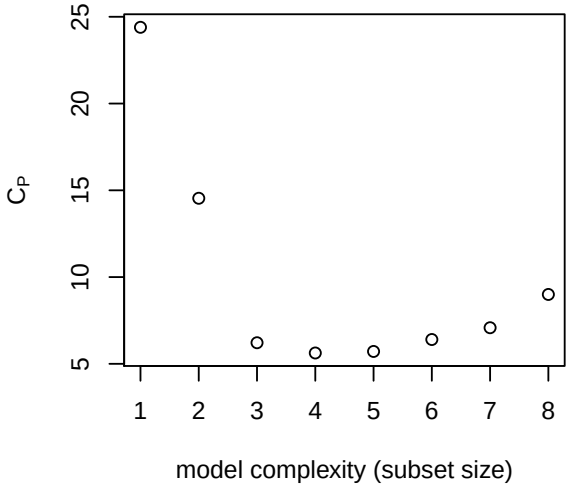
We can compare criteria, too. One way to do this is to re-run the `regsubsets` function to give only one best model per size (and one value of each criteria per size, which makes for easy comparison via plots). (We could also extract these values from our existing `prsub.sum` or `pstate.cri` objects, created above, but I find re-running easier.) The next chunk compares SSE, R^2_a, C_P (AIC) and BIC.

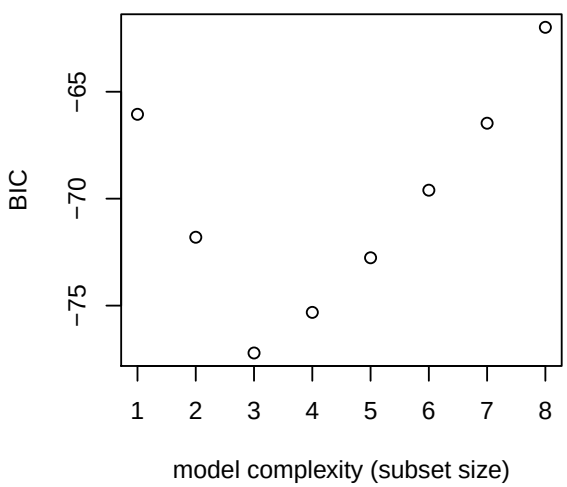
```
> p1best.sum<- summary(p1best.rbest<-
+                          regsubsets(lpsa ~ ., data=Prostate))
> ## summary(p1best.rsub) ## not run
> par(mfrow=c(2,2))
> plot(p1best.sum$rss,
+      xlab="model complexity (subset size)",
+      ylab="SSE")
> plot(p1best.sum$adjr2,
+      xlab="model complexity (subset size)",
+      ylab=expression(R[a]^2))
```

```
> plot(p1best.sum$cp,
+      xlab="model complexity (subset size)",
+      ylab=expression(C[P]))
> plot(p1best.sum$bic,
+      xlab="model complexity (subset size)",
+      ylab="BIC")
```









```
> par(mfrow=c(1,1))
```

Clean up.

We saw best model, in terms of BIC, in one of the above plots, but the some plots may be difficult to see which is the absolute best. Try this.

```
> detach(package:leaps)
> detach(package:lasso2)
```

3.5 Stepwise Procedures

- In addition to the exhaustive best subsets search procedure, illustrated above, there are various so-called stepwise-procedures. See, e.g., [KNNL05, §9.4]. Two of the simpler stepwise procedures, as far as I can determine, are briefly described below; see [HTF01, §3.3.1].
- Note that these procedures may be viewed as **greedy algorithms**, which make an "optimal" choice **at each step** (according to a chosen criterion) but are not guaranteed to find the best overall subset as an exhaustive search.
- On the other hand, as we said, an exhaustive search may be computationally infeasible.
- It appears that the `regsubsets` function in `leaps` package implements some version of these algorithms, but limited documentation makes me uncertain of the details without further investigation.
- See also the functions `add1`, `drop1` and `step` in the `R stats` package, and the functions `addterm`, `dropterm` and `stepAIC` in the `MASS` package.

- Again, a reminder that we're in the middle of relatively classical statistical selection procedures on our way to more modern ML procedures.
- We only mention these step-wise procedures, without illustration.

3.5.1 Forward-Stepwise Selection

- Typically, by default, this algorithm starts with the intercept parameter, then sequentially adds the input variable that produces the best improvement in model fit, by some criterion, thus producing a sequence of models of various sizes.
- Thus procedure can be used even when $p \gg n$ (**high dimensional data**). Why?

3.5.2 Backward-Stepwise Selection

- Typically, by default, this algorithm starts with the model containing all inputs, then sequentially eliminates the input that impacts the fit the least, by some criterion, thus producing a sequence of models of various sizes.
- Can only be used when $n \geq p$. Why?

3.6 Validation Set Approach

- Given that the primary **goal** of supervised machine learning is to **predict** one or more unobserved outputs, y, given corresponding,

previously unobserved input values, $\mathbf{x}$, that are not restricted to our original n observed values, but values that come from the same (perhaps vaguely defined) population of interest, the LS criterion somehow does not seem appropriate.

- After all, LS tells us to choose a function, $\widehat{\mu}$, to minimize sum-of squared errors, $(y_i - \widehat{\mu}(\mathbf{x}_i))$, $i = 1, \ldots, n$, as if we were somehow interested in predicting only the n outputs that we have already observed!
- Shortly, we will illustrate more concretely why LS is not an appropriate criterion for our prediction goal.

Alternatively, we consider choosing $\widehat{\mu}$ to minimize **generalization error**,

$$\mathrm{E}[(y - \widehat{\mu}(\mathbf{x}))^2 \,|\, \mathcal{T}], \tag{3.1}$$

where $\mathcal{T}$ is merely shorthand to denote our observed data set, i.e., $\mathcal{T} = \{(y_i, \mathbf{x}_i)\}_{i=1}^n$. ($\mathcal{T}$ for **training (data) set**; more on terminology as we go.)

- Expression (3.1) is a criterion that says that we want to choose $\widehat{\mu}$ to minimize error, on average, over the joint distribution of y and $\mathbf{x}$ (i.e., over the population of possible values of a single future observable datum pair), *given*, or *conditional upon*, what we have actually observed in our training data, $\mathcal{T}$, which we use to obtain, or train, the estimator/learner $\widehat{\mu}$.
- In other words, **if we had a second, very large data set** from the same population as, but collected independently from, $\mathcal{T}$, then we want to choose among such fitted $\widehat{\mu}$ the one that minimizes the average error over this larger, independent set; this second set is related to what we will call the **validation set**, shortly.

- Expression (3.1) is given in [HTF01, Expr. (7.2)]; we use squared error loss ([HTF01, Expr. (7.1)]).
- The LS criterion, on the other hand, appears somewhat analogous to (3.1), but tells us to choose $\widehat{\mu}$ to minimise SSE or, equivalently, to minimize an MSE,

$$\frac{\sum_{i=1}^{n}(y_i - \widehat{\mu}(\mathbf{x}_i))^2}{n}. \tag{3.2}$$

- This is called **training error** ([HTF01, Expr. (7.4)]), squared error averaged over our training data. Also, we often use the term **training MSE**.
- But, as we said, this LS minimization is only for our observed data, in $\mathcal{T}$, which seems to ignore our more general goal of prediction for values from the general population, not in our training data.
- (We should begin to see how this new criterion is getting at prediction beyond our training data, i.e., at generalizing to a large set ("population") of data.)
- We will tend to use the term **generalization error** to refer to (3.1), though **conditional generalization error** may be better (as our simulation example below suggests).
- We will see that this different goodness criterion generally results in the selection of "good" functions compared to the LS criterion (SSE or MSE) (for prediction, anyway, which is our ML goal).

- Above, we introduced the notion of **a second data set**, independent from the first, but from the same population.
- Usually, in practice, the size, n^*, of this second set, cannot be so large as to approximate precisely generalization error, as suggested above;

we will almost never be able to compute such average error over an entire population of values.

- As may be obvious by now, to distinguish the two data sets under consideration, we call our original data, $\mathcal{T} = \{(y_i, \mathbf{x}_i)\}_{i=1}^{n}$, a **training set** (or **training sample** or **model-building set**).
- We call the second set a **validation set** (or **validation sample** or **test set**, or similar), and will denote these data as $\mathcal{V} = \{(y_i^*, \mathbf{x}_i^*)\}_{i=1}^{n^*}$.

- It should be intuitively appealing to use the second, validation set to estimate generalization error (3.1), for a (learned/fitted) $\widehat{\mu}(\mathbf{x})$, with a **mean square prediction error**,

$$MSPR = \frac{\sum_{i=1}^{n^*}(y_i^* - \widehat{\mu}(\mathbf{x}_i^*))^2}{n^*}, \tag{3.3}$$

  where $\widehat{\mu}$ has been estimated from the separate, independent training set ([KNNL05, §9.6]).
- For slightly more insight as to why (3.3) may be a good estimator of (3.1), consider that $(y_i^*, \mathbf{x}_i^*)$ are (assumed to be) identically distributed across i^* (according to the same population distribution from which the training data is obtained).
- Conditioning on the training data, each random quantity, $(y_i^*, \mathbf{x}_i^*) \,|\, \mathcal{T}$, is identically distributed, so that, for each i^*, $\mathrm{E}((y_i^* - \widehat{\mu}(\mathbf{x}_i^*))^2 \,|\, \mathcal{T})$, is a constant, the same as our generalization error given in Equation (3.1).
- Thus, **MSPR** (3.3), when compute with a second, independent validattion set, is the average of such unbiased errors (squared) so is itself an unbiased **estimator of our generalization error**, which seems appealing.

- NOTE: We will discuss other methods, which attempt to estimate generalization error without a second, independent validation set; it is not so clear that these other methods estimate generalization error, as we will discuss a bit more later.

3.6.1 Validation Set Simulation Example

- We use a simulation example to **compare generalization error** (as estimated by MSPR) and **training MSE** (training error) in the context of model (variable) selection (i.e., selecting a $\widehat{\mu}(\mathbf{x})$, which we hope is somehow close to the actual $\mu(\mathbf{x})$, defined via minimizing a certain theoretical "expected prediction error" over the popuation of $(y_i, \mathbf{x}_i)$—see [HTF01, §2.4]), thus illustrating why generalization error is preferable (for prediction) to MSE obtained by LS fits to training data.
- Note that the example is similar to the prostate example, above, with 8 inputs. Thus, there are $2^8 = 256$ possible models (or 255 if we do not consider the model without covariates) for which to compute generalization error and training MSE.
- Though this is not a lot by today's standards, we will reduce the number models to 8, by using SSE to choose the best model of each size. As we said above, in the prostate e.g., our selection criteria are equivalent to SSE **for models of the same size**, though we have not made clear that SSE is/is not equivalent to generalization error for models of the same size.
- Still, we only compare generalization error and training MSE for these 8 models.
- We will see that the plot of generalization error vs. model size looks similar to plots of our selection criteria vs. size illustrated in our prostate example, above.

- This is no coincidence, as we will mention briefly, later.

Consider our model

$$y = \mu(\mathbf{x}) + \epsilon$$

where

$$\begin{aligned} \mu(\mathbf{x}) &= \mathbf{x}^t \boldsymbol{\beta} && (3.4) \\ &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 && (3.5) \\ &= 1 + x_1 - 0.5 x_2 + 1.5 x_3 - x_4, && (3.6) \end{aligned}$$

with

$$\epsilon \sim_{ind} N(0, 0.2^2),$$

and each of the four inputs are independently uniformly distributed on the unit interval, $[0, 1]$ ($\mathbf{x}$ is uniform on the unit hyper-cube $[0, 1]^4$).

- That is, for purposes of illustration, assume that we know the **true model**, including true function μ, from which data arise.
- Of course, in practice, we don't know our true model, and let's say that we will observe the **output**, the **four inputs** and **four additional inputs**, $x_5, \ldots, x_8$, each of which are also randomly distributed on $[0, 1]$.
- But, note that these **additional four inputs are not associated** with the output y because they're not part of our true model.
- Again, this simulation **scenario is similar to that of the prostate data, discussed above, with eight inputs** in total.
- We simulate $N = 100$ **training sets**, each consisting of $n = 50$ records, each record consisting of an output and eight inputs.
- For each training set, we obtain best models/subsets, one best model of each size subset using SSE, as we mentioned at the beginning of

this subsection. Thus, **for each of the $N = 100$ training sets, we will have 8 best models, one for each size**.

- And, for each training set/size, we compute **training MSE** for the best model, thus we have 8 training MSE values corresponding to 8 best models, one of each size. That is, 100 sets of 8 values.
- Further, we compute an ***average* training MSE** over the $N = 100$ simulated training sets, for each size, to get an idea of how this criterion might work in repeated application of our procedure. That is, 8 values of average training MSE.
- Also, we generate a single, largish **validation set** of $n^* = 200$ new values containing output y and the 8 inputs.
- Like for training MSE, 8 estimates of **generalization error**, one for each model size are computed. Again, 100 sets of 8 values.
- And, like for training MSE, of we average over the $N = 100$ simulated training sets, for each size, to get an idea of how this criterion might work in repeated application of our procedure. Again, 8 values, now of ***average* generalization error**.
- Follow the sequence of code chunks, below, with expanded discussion.

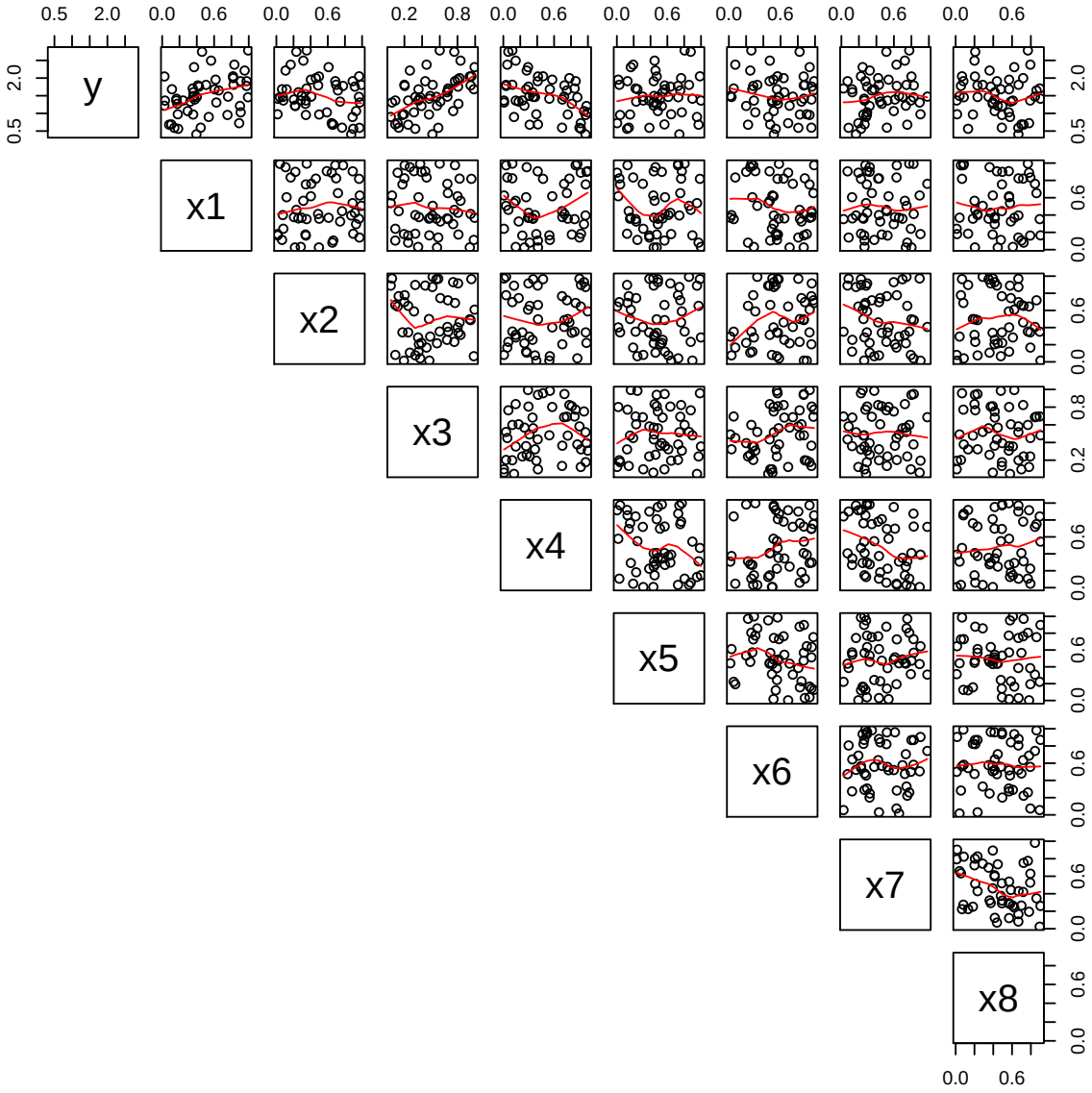
- The **next chunk generates N= 100 replicates of training sets**, each of size $n = 50$ with $k = 8$ potential inputs, four related to the output, four unrelated, according to the above linear model.
- In practice, we would typically observe only one training set of size n, of course.

```
> ## Generate 100 training sets each of size n=50.
> N<- 100
> n<- 50
> k<- 8
> btrue<- c(1, 1, -0.5, 1.5, -1) ## true mean parameter
> sig<- 0.2 ## true error variance is 0.2^2
> set.seed(24601 + 20500) ## Jean Valjean in the White House.
>
> ## Inputs
> Xlist<- sapply(1:100, FUN=function(x,n,k) {
+     res<- matrix(runif(n*k), nrow=n, ncol=k)
+     colnames(res)<- paste0("x",1:8)
+     res
+ },
+ n=n, k=k, simplify=FALSE)
> is.list(Xlist)

[1] TRUE

> length(Xlist)

[1] 100

> dim(Xlist[[1]])

[1] 50  8

> colnames(Xlist[[1]])

[1] "x1" "x2" "x3" "x4" "x5" "x6" "x7" "x8"

> ## Outputs
> ylist<- sapply(Xlist, function(x, n, btrue, sig) {
+     ytrue<- btrue[1,] + x[,1:4] %*% btrue[2:5,] +
+         rnorm(n=n, m=0, sd=sig)
+     },
+     n=n,
+     btrue=matrix(btrue,ncol=1),
+     sig=sig,
+     simplify=FALSE)
> is.list(ylist)

[1] TRUE
```

```
> length(ylist)

[1] 100

> head(ylist[[1]]); tail(ylist[[100]])

          [,1]
[1,] 0.9752157
[2,] 1.4681629
[3,] 1.4154548
[4,] 1.6842032
[5,] 0.7226639
[6,] 1.9984820
           [,1]
[45,] 2.2328183
[46,] 1.5540445
[47,] 2.1396520
[48,] 0.5824388
[49,] 0.5509606
[50,] 1.1485777

> ## Consolidate inputs and outputs
> Xylist<- sapply(1:100, function(x,Xlist,ylist)
+     cbind.data.frame(Xlist[[x]], y=ylist[[x]]),
+     Xlist=Xlist,
+     ylist=ylist,
+     simplify=FALSE)
> is.list(Xylist)

[1] TRUE

> dim(Xylist[[1]])

[1] 50  9

> is.data.frame(Xylist[[1]])

[1] TRUE

> names(Xylist[[1]])

[1] "x1" "x2" "x3" "x4" "x5" "x6" "x7" "x8" "y"
```

- Let's **look at one** of the $N = 100$ simulated training sets as an example of what we would see in practice, and as a sort of sanity check to make us feel better about not committing any gross errors when simulating data.
- The **first chunk below creates a plot of the first simulated training set**, and the **second chunk shows results from an LS fit** to the true model, which we do not know in practice, as a sort of brief sanity check for generated data.

```
> pairs(y ~ ., data=Xylist[[1]],
+       lower.panel=NULL,
+       upper.panel=panel.smooth)
```




```
> lm(y ~ x1 + x2 + x3 + x4, data=Xylist[[1]])


Call:
lm(formula = y ~ x1 + x2 + x3 + x4, data = Xylist[[1]])

Coefficients:
(Intercept)           x1           x2
     1.0677       0.9646      -0.4685
```

```
         x3          x4
     1.3884     -1.0593
```

- Now, we use the `leaps` package for each of the $N = 100$ training sets. By default, the procedure only returns one best model (`nbest=1`) of each size, thus giving $k = 8$ models for each of the $N = 100$ sets, as we said we would use.

```
> best.list<- sapply(1:100, function(x, Xylist){
+     require(leaps)
+     summary(regsubsets(y ~ ., data=Xylist[[x]]))
+ },
+ Xylist = Xylist,
+ simplify=FALSE)

Loading required package:  leaps

> length(best.list)

[1] 100

> names(best.list[[1]])

[1] "which"  "rsq"    "rss"    "adjr2"  "cp"
[6] "bic"    "outmat" "obj"
```

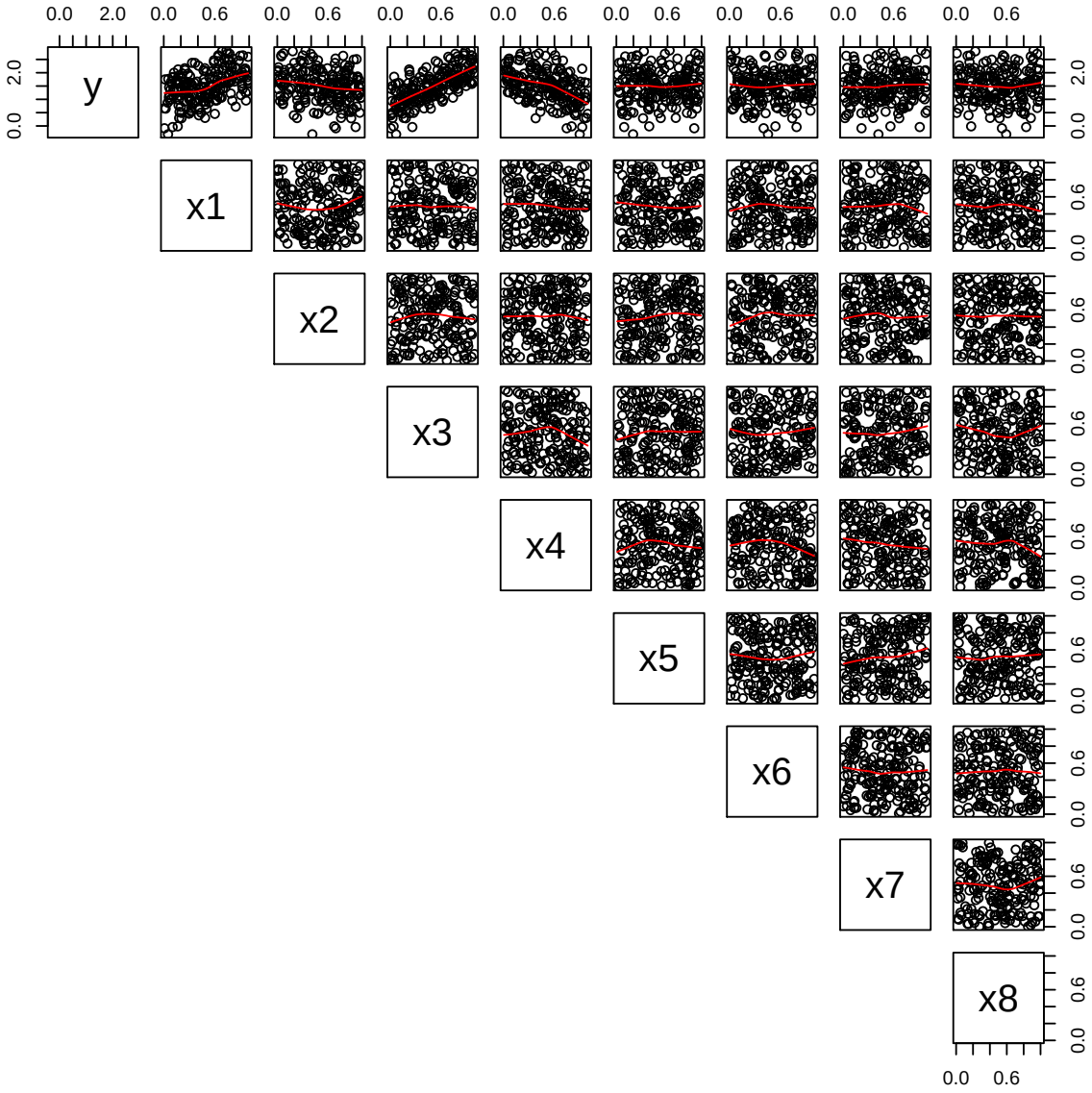
- Thus, contained in the above results from `leaps`, we have our 8 estimated models, $\widehat{\mu}$, for each of our $N = 100$ data sets. Again, 100 sets of 8 models. We'll return to these $N = 100$ `leaps` objects, shortly, to get and use our fitted models and training MSEs. xw
- First, we generate our one largish **validation set**, of size $n^* = 200$, in the same manner as we did the training sets; **mimicking the ideal**

that the validation set should be an independent set from the same population as the training set.

- The **next three chunks** show the code to simulate the validation data, plot the data, and check an LS fit, like our sanity check of (one of) the simulated training sets, above.

```
> ## Generate largish validation set of size n*=200
> nstar<- 200
> set.seed(24601 + 20500 + 777)  ## Lucky Jean-Valjean in the White House.
> XVmat<- matrix(runif(nstar*k), nrow=nstar, ncol=k)
>     colnames(XVmat)<- paste0("x",1:8)
> ytrueV<- btrue[1] + XVmat[,1:4] %*% btrue[2:5] +
+     rnorm(n=nstar, m=0, sd=sig)
> XyVmat<- cbind.data.frame(XVmat, y=ytrueV)
> names(XyVmat)

[1] "x1" "x2" "x3" "x4" "x5" "x6" "x7" "x8" "y"

> dim(XyVmat)

[1] 200   9
```

```
> pairs(y ~ ., data=XyVmat,
+       lower.panel=NULL,
+       upper.panel=panel.smooth)
```




```
> lm(y ~ x1 + x2 + x3 + x4, data=XyVmat)


Call:
lm(formula = y ~ x1 + x2 + x3 + x4, data = XyVmat)

Coefficients:
(Intercept)           x1           x2
     0.9495       1.0045      -0.5069
```

```
         x3          x4
     1.4747     -0.8224
```

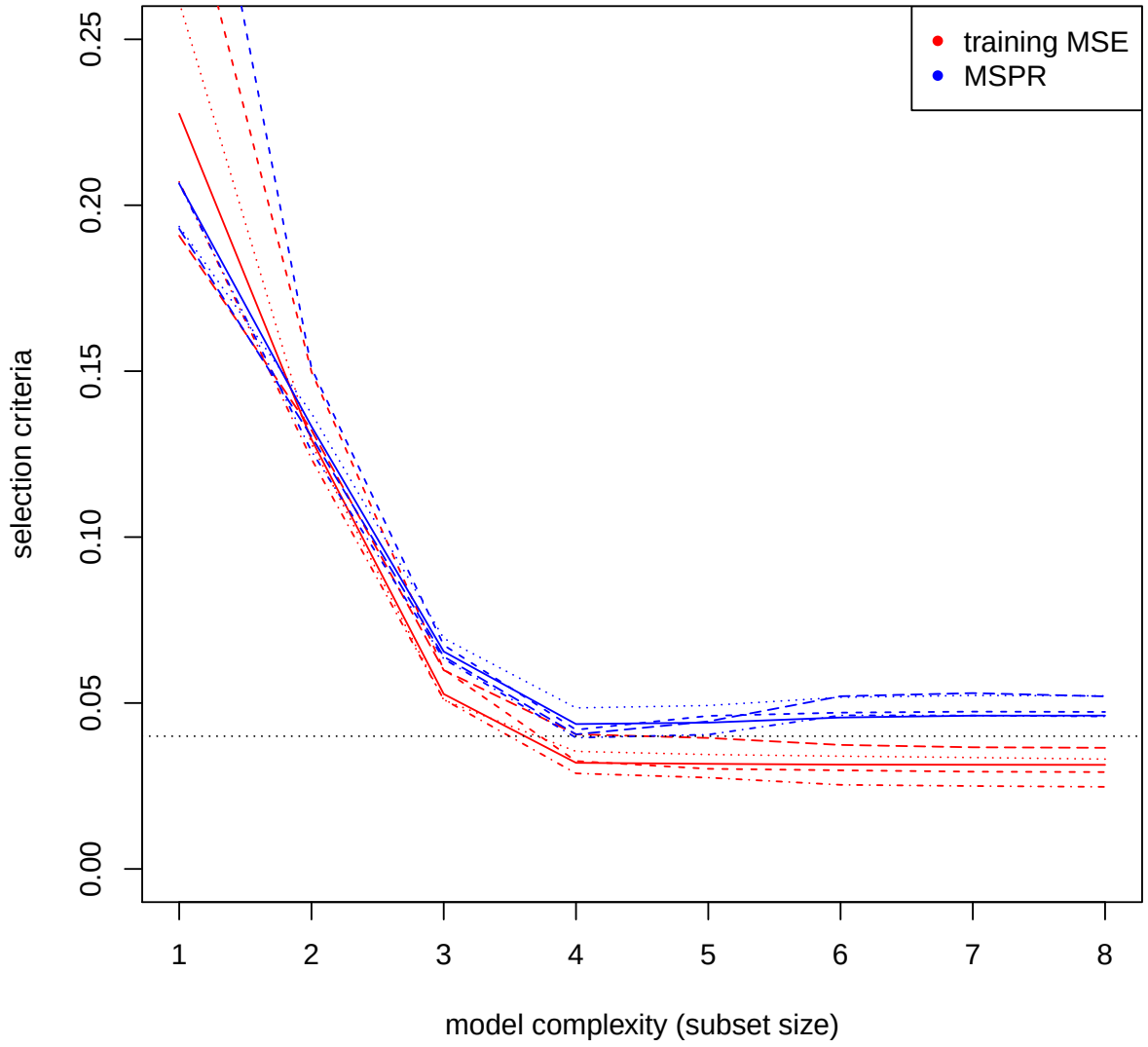
- Now we are ready use each of our 800 models to predict using the validation set. The first chunk below extracts the 800 fitted models into a list of length 100, each list component containing the 8 models for a training data set. We look at the first few models fit to the first simulated training set just to get familiar with things.
- Then, the chunk continues to compute the corresponding 800 $MSPR$ values. (We will obtain the training SSE hence training MSE values directly from the `leaps` objects; already computed.)

```
> ## N*k = 800 best fitted models
> best.mu.list<- sapply(best.list, function(x)
+     coef(x$obj, id=1:8),
+     simplify=FALSE)
> ## Just looking (more in class):
> length(best.mu.list)

[1] 100

> length(best.mu.list[[1]])

[1] 8

> best.mu.list[[1]][1:4]

[[1]]
(Intercept)          x3
  0.9543386   1.0783320

[[2]]
(Intercept)          x3          x4
  1.3406413   1.2630185  -0.9909757

[[3]]
```

```
(Intercept)          x1          x3          x4
  0.8666524   0.9311317   1.3835344  -1.0889151

[[4]]
(Intercept)          x1          x2          x3
  1.0676843   0.9645537  -0.4685395   1.3883879
         x4
 -1.0592911

> names(best.mu.list[[1]][[4]])

[1] "(Intercept)" "x1"          "x2"
[4] "x3"          "x4"

> ## For each of the N=100 training sets (outer sapply), predict the
> ## nstar=200 outputs of the validation set using the k=8 models and
> ## compute the corresponding k=8 MSPR values (inner sapply)
> mspr<- sapply(1:100, function(x, best.mu.list, XyVmat)
+     sapply(best.mu.list[[x]], function(bhat, XyVmat) {
+         X<- as.matrix(XyVmat[,names(bhat)[-1], drop=FALSE])
+         B<- as.matrix(bhat[-1])
+         yhat<- as.vector(bhat[1] + X %*% B)
+         mean((XyVmat[,"y"] - yhat)^2)
+     },
+     XyVmat = XyVmat),
+     best.mu.list = best.mu.list,
+     XyVmat = XyVmat, simplify=FALSE)
> length(mspr)

[1] 100

> length(mspr[[1]])

[1] 8
```

- The next plot shows the 8 values of **generalization error** ($MSPR$), connected by blue line segments, and, for comparison, the corresponding 8 values of **training** MSE, with red line segments for each of the first 5 training data sets. This gives an idea of the variability of these estimates over training sets. (Plotting all 100 looks a bit messy).

```
> ## MSE for first 5 simulated training sets
> plot(best.list[[1]]$rss / n,
+      ylab="selection criteria", xlab="model complexity (subset size)",
+      ylim=c(0,0.25), type="l", lty=1, col="red")
> invisible(sapply(2:5, function(x, bs, n) {
+     lines(bs[[x]]$rss / n,
+           lty=x, col="red")
+ },
+ bs=best.list,
+ n=n))
>
> ## Add corresponding MSPR
> invisible(sapply(1:5, function(x, mspr)
+     lines(mspr[[x]],
+           lty=x, col="blue"),
+     mspr=mspr))
>
> ## sig^2
> abline(h=sig^2, lty=3)
> ##
> legend(x="topright", legend=c("training MSE", "MSPR"),
+        col=c("red","blue"), pch=20)
```



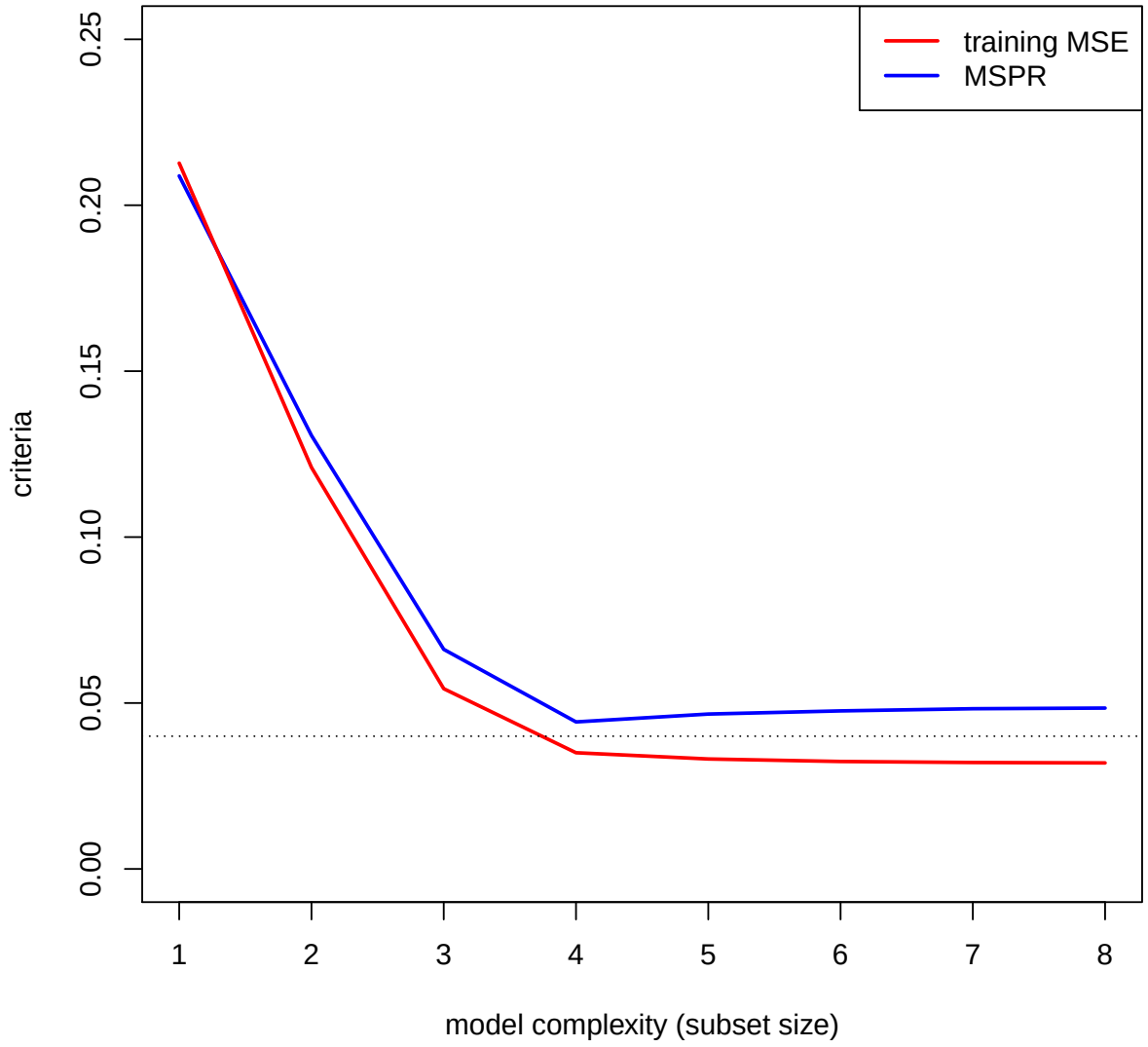
- We might be interested in how our procedure of using $MSPR$ (estimating generalization error) **performs in the the long run** so-to-speak, so we might average the $MSPR$ over our $N = 100$ training sets and compare to the corresponding average MSE.
- Incidentally, $MSPR$, *averaged over* $N = 100$ *simulated training sets*,

is likely a good estimate of our **generalization error** in (3.1) *averaged over the population of training sets* ([HTF01, Expr. (7.3)]).

- That is, we average MSPR over a (simulated) sample of $N = 100$ training sets, which estimates **expected generalization error** (expectation of (3.1)), where the expectation (average) is over the entire population of training sets of size $n = 50$ (and is expressed in [HTF01, Expr. (7.3)]; we use squared error loss).
- Thus, as mentioned above, we might use the term **conditional generalization error** for (3.1) and **expected generalization error** for its expected value ([HTF01, Expr. (7.3)]), but we continue to use **generalization error** in recognition of practice that tends to ignore such distinctions, despite many estimators being better at estimating **expected generalization error** ([HTF01, Expr. (7.3)]) rather than our preferred **(conditional)** generalization error ((3.1) & [HTF01, Expr. (7.2)]). (Using a separate, independent validation set, however, $MSPR$ is unbiased for (conditional) generalization error; as mentioned, other methods, which do no use a separate validation set, may estimate the expected generalization error better.)
- Accordingly, the next chunk shows a plot of the **average** $MSPR$, connected by blue line segments along with the corresponding **average** MSE values, connected by red line segments. Discussion in class, of course.

```
> ## average generalization error (average MSPR)
> mspr.mat<- sapply(mspr, I)
> dim(mspr.mat)

[1]   8 100

> mMSPR<- apply(mspr.mat, 1, mean)
> plot(mMSPR, type="l", lwd=2, col="blue", ylab="criteria",
+      xlab="model complexity (subset size)", ylim=c(0,0.25))
```

```
>
> ## average MSE
> mse.mat<- sapply(best.list, function(x,n) x$rss/n, n=n)
> dim(mse.mat)

[1]   8 100

> mMSE<- apply(mse.mat, 1, mean)
> lines(mMSE, lwd=2, col="red")
>
> ## sig^2
> abline(h=sig^2, lty=3)
> ##
> legend(x="topright", legend=c("training MSE", "MSPR"),
+        col=c("red","blue"), lty=1, lwd=2)
```



We notice a few things.

- Both (average) training MSE (average "training error") and (expected) generalization error (as estimated by (average) MSPR) correctly decline for models that are **too simple**, with **too few inputs** ($k < 4$).

- But, the **training MSE** of the best model of each size **continues to decline** with model size, so, strictly speaking, choosing the "best" model according to minimum training MSE would have us choose the model with all 8 inputs, despite, we know, four of the inputs being noise.
- **Generalization error**, however, reaches a minimum at $k = 4$, and **increases** for models that are **too complex** ($k > 4$), thus correctly identifies the best model (size anyway)(if we looked, we'd find that that the best model is the one with the first four inputs, our true model).
- That is, **generalization error penalizes models that are too complex** ($k > 4$), while **training MSE does not penalize complex models**.
- This makes training MSE (essentially, the least squares criterion) a poor model selection criterion, when left alone with a machine to do prediction for us, at least.
- We also notice **training MSE underestimates generalization error**; and, because of this, we often hear that least squares **overfits** models to the training data.
- Finally, notice that the plot of the (averaged) **generalization error vs size** looks similar to the shape of the plots of **selection criteria vs size** in the prostate example: down, then up (slightly, admittedly, for the simulated data). We'll return to this shortly.
- Incidentally, our previous two figures, when combined, are similar to [HTF01, Fig. 7.1] (which uses LASSO, not best subset selection).

So, what *is* the best selected model? This depends on the training data set, of course, and we have only one of these, in practice. Our plots

suggest a model (variable susbset) size of 4. Which 4 inputs? Does our final, selected fitted model, $\widehat{\mu}(\mathbf{x})$, match our true model, $\mu(\mathbf{x})$? Of course, we don't know $\mu(\mathbf{x})$ in practice, linear or non-linear, but we hope somehow that our fitted model, $\widehat{\mu}(\mathbf{x})$, is some how close to the true model. (As briefly mentioned above, look at [HTF01, §2.4] for what sorts of $\mu(\mathbf{x})$ that we might hope to be close to.) Let's look at the best fitted model for a few training sets.

```
> set.seed(5551212)
> sample(1:N, size=3)

[1] 64 47 79

> dim(mspr.mat)

[1]   8 100

> which.min(mspr.mat[,64])

[1] 4

> best.mu.list[[64]][which.min(mspr.mat[,64])] ## got truth!

[[1]]
(Intercept)          x1          x2          x3
  1.0596489   1.1061974  -0.6426814   1.4231765
         x4
 -0.9796132

> which.min(mspr.mat[,47]) ## size 4

[1] 4

> best.mu.list[[47]][which.min(mspr.mat[,47])] ## got it!

[[1]]
(Intercept)          x1          x2          x3
  1.1469776   0.9064649  -0.5362310   1.4019634
         x4
 -1.0041247
```

```
> which.min(mspr.mat[,79]) ## size 4

[1] 4

> best.mu.list[[79]][which.min(mspr.mat[,79])] ## got it!

[[1]]
(Intercept)          x1          x2          x3
  0.9857183   0.8783477  -0.3437992   1.3994393
         x4
 -0.9741350
```

3.7 Model Assessment

- So far, we have been concerned with **selecting** the best model, in particular, the best subset of inputs, where bestness is determined by generalization error, as estimated by $MSPR$.
- As it turns out, **the process of selecting a best model from many candidate models tends to result in an estimated model that under estimates generalization error**. (We did not see this in our simulation example with our best (average) MSPR being a bit higher than $\sigma^2 = 0.2^2 = 0.04$.)
- This does not tend to be a problem if selecting the best model is our end goal, as I will attempt to explain in a non-technical way, shortly.
- But, if we want to conduct **inference** using, e.g., standard errors, p-values, error rates and interval coverage rates, then underestimation of generalization error will affect these quantities.
- In particular, a model that underestimates generalization error will tend to cause inferences to be **optimistic or over-stated**. For example, confidence intervals will be too precise for their nominal coverage rate, or, equivalently, the actual coverage rate (confidence) will be less than stated.

- You might say that machine learning is not ordinarily concerned with such traditional statistical activities and conclude this is not an issue. But, if you want to construct valid *prediction* (or confidence) intervals, with actual coverage rates matching nominal (stated) coverage rates, then this underestimation does come into play in machine learning.
- Thus, we distinguish **two goals** ([HTF01, §7.2]):
  - **Model selection**: Estimate model performance (e.g., generalization error) of different models for the purpose of choosing the best.
  - **Model assessment/confirmation**: Estimate performance (e.g., generalization error) for the purpose of obtaining valid inference, e.g., prediction errors and intervals
- In order to discuss **model assessment**, we mention a **third data set** that arises in the current validation set approach: the **test set** (or **confirmatory set**). See the nearby figure illustrating the ideal scenario of having all three such data sets (Source: [HTF01]).

- **To explain further**, suppose we somehow have **100 equally good models for predicting** a new y using a new $\mathbf{x}$ *obtained from the population* of such $(y, \mathbf{x})$ (only x observed).
- But, the **validation set**, like the training set, is a relatively **small subset of the larger population** of such values. Thus, while each model is equally good when validated against then entire population (we can't do this in practice, of course), the variability of observations in the population will result in a particular validation set that now fits one model "best" by **chance alone**, and, in this sense, using the same validation set to choose among several models results in a model that is **overfit to the validation set**, resulting in an estimated generalization error that tends to be too small.

- Of course, we can imagine that we are not selecting among 100 equally good models, but among models that do actually differ in their prediction goodness on the population.
- Thus, if models are different enough, **the *selection* process tends to overcome chance to reveal the true best model**.
- And, if models are not different enough to be distinguished from chance, then we may not care (for prediction purposes) which model is selected. Thus, as far as the goal of *selection* goes, we tend to get the best model or something close to it.
- However, if we want **assess** the final selected model or use that model to estimate predict error, we should use the model to predict on the **test set** and hence to compute a fresh estimate of generalization error that better characterizes the error with which we predict construct prediction intervals.
- Also, we may fit the chosen model to the **test set** to get more realistic inference (p-values, etc.) for parameters or for predictions.
- Thus, we use the validation set for **selection** of the best of many models, and the **test set**, if desired, is used for **assessment** of the final model with (hopefully) accurate p-values, etc.

3.8 Alternatives to the Validation Set Approach

- An **independently collected validation set** using $MSPR$ for model selection is an ideal that **may not be achievable in practice**; we may not have a second set or, **splitting our data into two sets** may result in small sets (training or validation) that suffer from too little information to get reliable estimates of μ or reliable estimates of generalization error.

- Thus, there are **two popular alternatives to the validation set approach** that do **not require a separate validation set**.
  - (1) Somehow **adjust training MSE** to account for model complexity.
  - (2) **Cross-validation** attempts to estimate the generalization error directly, without attempting to adjust training MSE.
- Still, these alternative methods, like the validation approach, tend to result in a final estimated model that underestimate generalization error when used to select among many models. Thus, the above discussion of **model assessment still applies**, though the **test set** is a second set, not third, for the alternative approaches, of course.

3.8.1 Adjustments for Model Complexity

- It turns out that the criteria, AIC (C_P) and BIC, implement approach (1), adjusting training MSE for model complexity in an attempt to **approximate generalization error**.
- Thus, inasmuch as generalization error is an intuitive goodness criterion for model selection, these previously somewhat mysterious criteria should now be seen less mysteriously for selecting models.
- In particular, as alluded to, above, we now have some understanding why the plots of **selection criteria vs. size**, for the prostate data, look similar to the plots of **generalization error vs. size**, for the simulated data.
- Again, this approximating approach **does *not* use a separate validation set**, though we may still want the above mentioned test set (second set, in this case) for assessment of the final selected model.

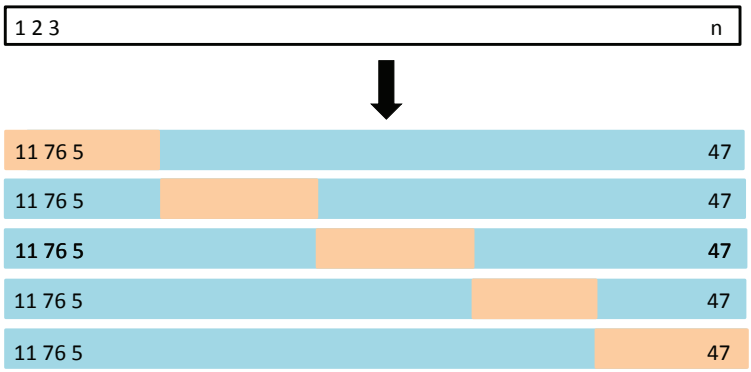
- Thus, we just fit all models to the training data, compute our favorite information criterion, and choose the best model according to the minimum AIC for example. How these criteria are/are not an estimate of generalization error (3.1) is relatively technical; see [HTF01, Ch. 7].
- We've already seen an example of this adjusting approach (1) in the above prostate example illustrating the best subsets approach. No validation set used.
- We tend not to consider further the test set.

3.8.2 K-fold Cross-Validation

- The second alternative approach (2) to the validation set approach is K-fold cross-validation (CV), which attempts to **mimic the collection of an independent validation set**.
- Not having a second, validation set, consider the case wherein we randomly **split the original data set into a training set and a validation set**.
- This essentially gives an independent validation set, as discussed above, and we may proceed as above, according to the validation set approach, using $MSPR$ to estimate generalization error to selected models.
- But, as mentioned, **if the sets are too small**, then we may not get reliable results: **poor fits** to the training set and **poor estimates of generalization error** may make for poor model selection, not to mention assessment with a test set.
- In this case of randomly splitting your data set into two sets, **you might think to perform validation twice** by interchanging, or

crossing, the roles of the **training** set and the **validation** set in the hope of somehow improving on a single validation procedure by **averaging the resulting 2 estimates of generalization error**.

- Good idea. This is **2-fold cross-validation**.
- Extend this idea to K "folds:" split your data into K sets ("folds"), use $(K-1)$ sets as the (collectively single) training set and predict on the remaining fold, the validation set, and repeat K times, **averaging K estimates of generalization error** to get an overall estimate of generalization error ([JWHT14, Ch. 6], [HTF01, §7]).
- See the nearby simple diagram depicting 5-fold CV (Source: [JWHT14]).
- The extreme of **n-fold CV** considers (cross) validation sets with a single, $n^* = 1$, value to be predicted by models fit to training sets of size $n-1$.



- More precisely, we use the following **estimate of generalization error**

$$CV_{(K)} = \frac{1}{K}\sum_{i=1}^{K} MSPR_i,$$

where $MSPR_i$ is computed from the estimated model, $\widehat{\mu}_i$, fit to the training data set of fold i, to predict the outputs, y_j^*, in the validation set of fold i, using the inputs, $\boldsymbol{x}_j^*$, in the validation set of fold i, $j = 1, \ldots, n_i^*$. That is,

$$MSPR_i = \frac{\sum_{j=1}^{n_i^*} (y_j^* - \widehat{\mu}_i(\boldsymbol{x}_j^*))^2}{n_i^*}. \tag{3.7}$$

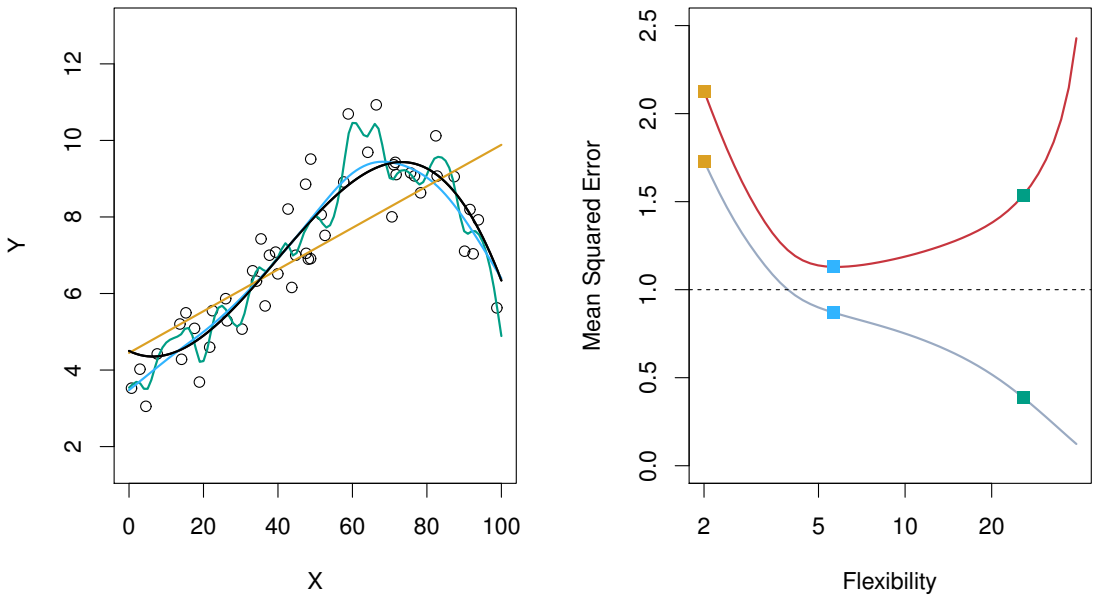
- See, e.g., [JWHT14, Expr. (5.3)], where their MSE_i is our $MSPR_i$.
- But, **technically, things are now different** from the relatively straightforward validation set approach.
- For example, we are now conditioning on **different training sets**, depending on which subset of folds is used for the training set and which is used for the validation set, so that, technically speaking, we would be estimating K **different generalization errors**, averaging these estimates according to the above $CV_{(k)}$ criterion, in the hope of getting at generalization error, Expression (3.1) (which is conditioned on a different set yet, the entire, single set).
- Also, the **training data is shared** across K so that estimates of the generalization error are **correlated across** K (though this latter point is evidently not true for $K = 2$ folds).
- In any case, the analysis of **what exactly is being estimated** (Equation (3.1) or something else) by $CV_{(k)}$ is relatively technical. See [HTF01, Ch. 7], in particular [HTF01, §7.12].
- Still, $CV_{(k)}$ appears to provide some sort of estimates of generalization error that do a *relatively* good job in the sense that the procedure has the reputation of resulting in a model close to the best model, even if the estimates of generalization error are not accurate in an *absolute* sense.

- Again, as already mentioned, whether using the validation set approach, or the alternative approach (1) that skips the validation set—approximating generalization error by adjusting the fitting procedure using just a single data set (e.g. AIC, BIC, C_P)—or the alternative approach (2), K-fold CV, the process of selecting the best model among many candidate models tends to result in the best selected model giving an **underestimate of generalization error**, as well as underestimates of standard errors for the more traditional endeavors such as hypothesis testing and interval estimation. Thus, we may want to consider using a **test set**, as mentioned.

CV Example. While K-fold CV may be used to select from several inputs, as in the above prostate example or the simulation example, as well as in more general model selection scenarios, it is often used to select a flexible (but not too flexible) model between an output, y, and a single input x.

- Using **splines**, the original input, x, is transformed effectively into multiple inputs (of the kind x^2 and x^3, roughly speaking) that then are used in a linear model fit; these new, transformed inputs fill the columns of the $\mathbf{X}$ in our linear model. We may learn more details about splines, later.
- The following figures (source: [JWHT14]) shows (left) simulated training data along with the true relation, μ, between y and x (**black**), and three fitted models, $\widehat{\mu}$: a simple linear regression fit (**orange**), and two spline fits (**blue** and **green**).
- The right figure shows **training MSE** and **generalization error** (which we will assume was estimated by K-fold CV), along with the particular values of the fits indicated by appropriately colored squares.

- **Model flexibility** (complexity/size) is familiar to us for the simple linear regression fit: $p = 2$ parameters (**orange** square), corresponding to the number of columns in the $\mathbf{X}$ matrix. Flexibility for the spline fits also refers to the **number parameters**/columns of their $\mathbf{X}$ matrices (**blue** square and **green** square), the details of which are left for later coverage of splines.
- Again, like our previous examples, we see, perhaps more clearly now, **overfitting** indicated by the departure of **training MSE** and **generalization error** in the right plot: training MSE does not penalize for model complexity, as we mentioned, above. But, now, this 1-d scenario (single original x input) lets us see overfitting perhaps more intuitively in the left plot.
- We assumed the plot of generalization error was produced using CV. Based on our previous discussion, we would this to look similar to plots of $MSPR$ vs. size, from a validation approach, or plots of a selection criterion vs. size.



3.9 Bias-variance Trade-off

- The so-called **bias-variance trade-off** is often discussed in conjunction with the over-fitting of models with ever-increasing complexity that is revealed by such a comparison of training MSE and (estimated) generalization error.
- We give only a brief, intuitive discussion of the bias-variance trade-off. We'll look at the plot of the previous CV Example as we go.
- Intuitively and loosely speaking, a relatively **simple** fitted model, $\widehat{\mu}(\mathbf{x})$, will not capture the complexity of the true relationship, $\mu(\mathbf{x})$, thus, will be **biased** for $\mu(\mathbf{x})$.
- Also, a **simple** fitted model will tend to have **low variability** as simple models are relatively easily informed by even relatively small data sets.
- **For example, a constant model**, $\widehat{\mu}(\mathbf{x}) = \widehat{\mu}$ will generally estimate the overall level of the outputs (i.e., the unconditional mean, $\mu = E(Y)$, not $\mu(\mathbf{x}) = E(Y|\mathbf{x})$) with high precision, or **low variance**, but will generally be **biased** for the conditional mean, our unknown regression function, $\mu(\mathbf{x}) = E(Y|\mathbf{x})$.
- As an estimated model **increases in complexity** from such a simple model (e.g., as the input vector $\mathbf{x}$ gets bigger, with more covariates), it is better able to capture the complexity of $\mu(\mathbf{x})$, and becomes **less biased**. But, at that same time, without an increasing n, such models exhibit **more variability**. Intuitively, we have **fewer data points per parameter** to be estimated, and this tends to result in the parameters hence model being fit with less precision/more variability.
- At some **intermediate point of model complexity**, model bias and model variance give a model that predicts well, as indicated by (estimated) generalization error.

- But, as the model becomes **even more complex**, the model gets too close to the training data, as the LS criterion suggests and as training MSE shows. Thus, the model tends to acquire the properties of the training data. In particular, while it becomes **less biased** for the true $\mu(\mathbf{x})$, ***at the training inputs***—the data y_i are unbiased for their mean, $\mu(\mathbf{x}_i)$, of course, by definition—the estimated model tends to exhibit the variability of the data that are too few to estimate such a complex model with high precision. And this **increased variability tends to swamp lower bias** as indicated by **increasingly poor prediction** with complexity beyond the intermediate model.

3.10 Summary

- In the case of linear models, different models arise from the choice of different inputs (covariates). In this case, **model selection** is often called **variable selection**. (This tends to be true for generalized linear models, too.) However, model selection is more general, e.g., we may select between a normal linear model and a log-normal model (though C_P is only for linear models).
- We mentioned that we could use a relatively **traditional statistical approach, using t or F tests**, to select among models, but we quickly dismissed this to move toward more modern, ML methods.
- We used **selection criteria** (AIC, etc.) in a **best subsets** (variable selection) procedure using the prostate data. In this case, our **search was exhaustive** over all possible models (all possible subsets of inputs (without transforming to create further inputs)). We saw a best model(s) emerge at an intermediate level of complexity (number of inputs). We mentioned step-wise procedures for situations where the number of models (variable subsets) are too numerous to enumerate and compute, but we did not pursue these step-wise procedures.

- Initially, we did not discuss how these selection criteria were related to our **goal of prediction**, and we proceeded to discuss the **validation approach** using a validation set to compute an estimate of **generalization error** (MSPR), which we should see as a performance criterion directly related to our goal of prediction. Using our simulation example, we illustrated how **training MSE** (from least squares), was **not appropriate** as a prediction performance criterion using a simulation example that was similar to the prostate example. And, we saw how plots of generalization error vs. size, in the simulation example, looked similar in shape to plots of a selection criteria vs. size, in the prostate example.
- Because a second, independent validation set may not be feasible, we discussed **alternatives to the validation approach**.
- It turns out that use of **selection criteria** (AIC, etc.) was one such approach, which somehow adjusts training MSE to better reflect generalization error. We avoided details, but, again, we now had some understanding of why selection criteria plots look similar to generalization error plots.
- We also discussed **cross-validation** as another alternative to the validation approach, again seeing plots of the CV criteria vs size being similar to previous methods' plots.
- In any case—validation approach, selection criteria, or CV—our selection procedures often result in **under estimation of generalization error**, which tends not to be such a problem for selection, but does present a problem if we want to conduct inference, such as constructing prediction intervals. In this case, we may want to consider a **test set** to obtain more appropriate inferences (p-values, coverages rates). We largely avoided model assessment with such a test set.
- Finally, we illustrated **overfitting** and the **bias-variance trade-off** using a (assumed) CV example from [JWHT14]. Looking back at our prostate data and similar simulation example, wherein we used

selection criteria, we can see, in those examples, too, the over-fitting and bias-variance trade-off in the plots.

Lecture 4

Bayesian Linear Model

Contents

Main Objectives:

- General Bayesian model
- Bayesian linear model
- Data distribution (likelihood), prior distribution, posterior distribution, Bayes theorem, distribution kernel, prior predictive distribution, posterior predictive distribution, conjugate prior, improper prior, full conditional distributions.
- Shrinkage

$\mathcal{O}$

Reading Assignment:

[Bis06, §3.3], [Mur12, §4.4, 4.6], [Wak13, §3.2, 3.4, 3.7, 3.8, 3.12, 4.4, 4.11, 5.12, 5.13]

INF511Notes.pdf (see specific references below) $\mathcal{R}$

4.1 Introduction

- Some fundamental results on distributions, including relatively generic presentations of joint, marginal and conditional distributions, probability density functions (pdfs) and probability mass functions (pmfs) are given in §5.2 of INF511Notes.pdf.
- Fundamental properties of distributions/random variables (expectation, variance and covariance), are given in Chapter 3 and §4.3 of INF511Notes.pdf.
- We recover only some of that material here, in more of a Bayesian context, similar to Chapter 8 of INF511Notes.pdf, though some changes have been made here.

4.2 (Un)Observed Data Distribution

- Let $\boldsymbol{Y}$ be a random vector of **observables** (responses/outputs) whose distribution depends on a vector of parameters, generically denoted as $\boldsymbol{\theta}$ for the moment.
- In anticipation of $\boldsymbol{\theta}$ being formally considered as a random variable in a Bayesian framework, we may write the random vector as $\boldsymbol{Y} \,|\, \boldsymbol{\theta}$ and denote its (conditional) distribution as

$$[\mathbf{y} \,|\, \boldsymbol{\theta}].$$

- We will sometimes use the term **data distribution** to refer to our model for our data.
- E.g., classical linear model, in class (and in subsequent sections)
- It's conventional, among Bayesian statisticians, to **explicitly condition on random variables** ($\boldsymbol{\theta}$ is now an rv...see below) and to

omit conditioning on fixed quantities, like covariate/input $\boldsymbol{x}$, in a linear model, but this convention is **not universally followed**, particularly outside of the Bayesian statistical literature, depending on context.

- Recall, if we view the **data distribution as a function of the parameters**, data held fixed at their observed values, we get the **likelihood** (function) of the data (§6.4 INF511Notes.pdf).
- Incidentally, we have the (log-)likelihood for a typical **normal** linear model in chapter 2 of our notes, and we have referred to **generic** likelihoods when discussing the selection criteria, AIC and BIC, which may be viewed as **penalized (log-)likelihoods**, in chapter 3.
- In the case that we want to predict unobserved data, we may specify a **joint distribution of unobserved and observed data**,

$$[\mathbf{y}^*, \mathbf{y} \,|\, \boldsymbol{\theta}] = [\mathbf{y}^* \,|\, \mathbf{y}, \boldsymbol{\theta}][\mathbf{y} \,|\, \boldsymbol{\theta}],$$

where we have used $\mathbf{y}^*$ to denote unobserved data (outputs) and $\mathbf{y}$ as observed data (outputs).

4.3 Random Parameter

- Until now, we have conceptualized the unknown parameters (e.g., $\boldsymbol{\beta}$ or σ^2) in our data distribution as **unknown but fixed** quantities to be estimated, with **uncertainty** in estimation following from **sampling distribution theory** (in simple cases, e.g. [Wak13, p. 29 & Ch. 5] or from **asymptotic theory** (e.g., much of [Wak13, Ch. 2]).
- See Chapter 6, INF511Notes.pdf, covering typical sampling distribution results, involving t and F distributions, for the normal linear model, some of which we recovered in Chapter 2 here in INF 504, very quickly.

- In a Bayesian analysis, we take a different tack, and we characterize our **uncertainty** (or **belief**) about unknown parameters more directly via a **prior distribution** (a marginal distribution) and a **posterior distribution** (a conditional distribution) for the parameter itself (rather than with the distribution of an estimator of the fixed parameter, as in our t and F material so far). See [Wak13, Ch. 3].
- In other words, **we may still conceptualize the parameters as fixed**, but our accounting of **uncertainty of the fixed quantities is formalized by probability distributions** for parameters. That is, we treat the parameters as random as a way to characterize our uncertainty about them.
- Like frequentist methods that use a data distribution, Bayesian methods also begin with a data distribution, now with the additional specification of a **prior distribution** for the unknown parameters,

  $$[\boldsymbol{\theta}],$$

  which embodies our beliefs (or uncertainty) about the parameters, before (ideally) looking at the observed data.
- Note that the prior distribution is a distribution as any other, with its own **(hyper) parameters** (implied and often not denoted), that must be specified with known values.
- In a Bayesian analysis, the goal is often to obtain the conditional distribution

  $$[\boldsymbol{\theta} \,|\, \mathbf{y}],$$

  known as the **posterior distribution**, which embodies our belief (or uncertainty) of the parameter after somehow updating our prior belief with observed data.
- Imagine having the distribution of your unknown quantities of interest—that accounts for the information in your data (via the likelihood as we will see) and for prior information (via the prior as we

will see)—for use in inferring about the unknown quantities given what you have observed. This seems like an agreeable object to have for making inference.

- Also, as considered briefly in the frequentist linear model (§6.7.9 INF511Notes.pdf, and, again, very quickly in Chapter 2 here), we may want to **predict** an as yet unobserved response/output (vector), $\mathbf{y}$, with less emphasis on parameter $\boldsymbol{\theta}$.
- We might simply include such unknowns to be predicted in $\boldsymbol{\theta}$ (see below), but we often give explicit recognition to these unknown quantities in the form of the **posterior predictive** distribution,

$$[\mathbf{y}^* \,|\, \mathbf{y}],$$

  where we use $\mathbf{y}^*$ to denote unobserved observables (e.g., responses/outputs we want to predict) and continue to use $\mathbf{y}$ to denote observed observables/responses/outputs, as in our (un)observed data distribution, above.
- Again, this seems to be a nice object to have for inferring about unknown quantities, given what you have observed.
- Generically, we might consider our goal to be the distribution of

$$[\mathit{unknowns} \,|\, \mathit{knowns}],$$

  which, given our discussion so far, we may denote as $[\mathbf{y}^*, \boldsymbol{\theta} \,|\, \mathbf{y}]$ (a **joint posterior** of sorts).
- We work our way to $[\boldsymbol{\theta} \,|\, \mathbf{y}]$, $[\mathbf{y}^* \,|\, \mathbf{y}]$ and $[\mathbf{y}^*, \boldsymbol{\theta} \,|\, \mathbf{y}]$, first, generically, in what follows, then in terms of the linear model, a bit later.

4.3.1 Bayes Theorem: Posterior $\propto$ Likelihood $\times$ Prior

- If we specify a (marginal) **prior distribution**

$$[\boldsymbol{\theta}],$$

then, with the data (conditional) distribution, we can **build a joint distribution, hierarchically, using the multiplication rule** (Definition 5.6 INF511Notes.pdf)

$$[\mathbf{y}, \boldsymbol{\theta}] = [\mathbf{y} \,|\, \boldsymbol{\theta}][\boldsymbol{\theta}].$$

- From this **joint** distribution, we can (*in principle*) get our desired (**posterior**) **conditional** distribution as

$$[\boldsymbol{\theta} \,|\, \mathbf{y}] = \frac{[\mathbf{y}, \boldsymbol{\theta}]}{[\mathbf{y}]}, \quad \text{or}$$

$$[\boldsymbol{\theta} \,|\, \mathbf{y}] = \frac{[\mathbf{y} \,|\, \boldsymbol{\theta}][\boldsymbol{\theta}]}{[\mathbf{y}]},$$

where we have simply used the generic definition of conditional distribution (§5.2 of INF511Notes.pdf). See also [Wak13, Expr. (3.1)].

- This result is known as **Bayes Theorem**
- We sometimes hear the term **inverse probability**, because, instead of computing probabilities of (functions of the) data given the parameter, as in a frequentist approach, we now compute probabilities of the parameter given the data.
- We'll discuss prior/posterior distributions, in the context of linear models, soon.
- Notice that the **marginal distribution** (**prior predictive**), $[\mathbf{y}]$, is a constant wrt $\boldsymbol{\theta}$ in the posterior distribution, thus it does not help to distinguish among different values of $\boldsymbol{\theta}$; it is not a necessary part of the **kernel** ("important factor") of the posterior distribution and is part of its **normalizing constant** (or may be referred to as *the* normalizing constant of the posterior [Wak13, Expr. (3.2)]).

Generically, the **kernel** of a distribution, $f(x) = cg(x)$, is the factor, $g(x) \geq 0$, that depends only on the random variable/vector, x, with the remaining multiplicative constant (wrt x), c, being the **normalizing constant** that results in total probability of 1; i.e.,

$$1 = \int f(x)\mathrm{d}x = \int cg(x)\mathrm{d}x \quad \text{gives}$$

$$c = 1/\int g(x)\mathrm{d}x$$

Thus, as long as we can somehow perform the integration of the kernel, we can recover the distribution from the kernel, and in this sense we see the kernel to be the **important part** of the distribution. It can be advantageous to recognize distributions by their kernels, in which case we can avoid integrating/summing to recover the distribution.

- **If we recognize the kernel of the posterior**, great! "Hey, that factor looks like it belongs to a normal distribution for $\boldsymbol{\theta}$ (up to a normalizing constant that does not depend on $\boldsymbol{\theta}$)!" In this case, we're done, up to summarizing our posterior, at least. We don't have to integrate (or sum; what?) to get the normalizing constant to get the posterior.
- In this case, we might write **Bayes Theorem** as

  $$[\boldsymbol{\theta} \,|\, \mathbf{y}] \propto [\mathbf{y} \,|\, \boldsymbol{\theta}][\boldsymbol{\theta}],$$

  which we may often hear as ([Wak13, p. 86])

  > the posterior is proportional to the likelihood times the prior.

- But, for all but the simplest situations, **we may not recognize the kernel** of the posterior. We may think then that we must somehow use numerical quadrature methods to integrate (sometimes very

high dimensional) or to otherwise numerically approximate integrals (see, e.g., [Wak13, §3.7]) in order to obtain $[\mathbf{y}]$ for use in our original statement of Bayes Theorem.

- We tend to focus more on Markov chain Monte Carlo (**MCMC**) methods, which effectively get around such integration. (We may view (MC)MC methods as integration methods, but, as typically implemented, this view is obscured.)
- That is, most MCMC methods do not require the normalizing constant, $[\mathbf{y}]$, to effectively reproduce the posterior $[\boldsymbol{\theta} \,|\, \mathbf{y}]$. (Poor normalizing constant, almost no one needs you...)
- Thus, again, we have the tendency to express **Bayes Theorem** as

$$[\boldsymbol{\theta} \,|\, \mathbf{y}] \propto [\mathbf{y} \,|\, \boldsymbol{\theta}][\boldsymbol{\theta}].$$

- **Classical Bayesian linear models are relatively simple**, and we often recognize (conditional) kernels to give us the posterior or give us full conditional posterior distributions which are a step toward the posterior.

4.3.2 Prior Predictive and Posterior Predictive Distributions

Now let's move toward the **posterior predictive**, $[\mathbf{y}^* \,|\, \mathbf{y}]$, in the case that we want to infer about unobserved responses/outputs $\mathbf{y}^*$ rather than or in addition to parameters, $\boldsymbol{\theta}$. Again, machine learners often do not predict unobserved random variables but, more often, the mean of such unobserved random variables. More as we go.

- The marginal distribution (normalizing constant of the posterior, above),

$$[\mathbf{y}] = \int [\mathbf{y} \,|\, \boldsymbol{\theta}][\boldsymbol{\theta}] \mathrm{d}\boldsymbol{\theta},$$

(or summing) is sometimes referred to as the **prior predictive distribution**, as noted above, because it is (in principle) the distribution we would use to infer about (predict) our data $\mathbf{y}$, **before we observe it**, after integrating/summing out, i.e., accounting for the uncertainty of the quantities we don't know, the unknown parameter $\boldsymbol{\theta}$.

- It is **sometimes used to assess models** (likelihood and prior) by comparing it to observed data. Intuitively, if we plug in our actual, observed data, $\mathbf{y}$, into $[\mathbf{y}]$, and we get an unusually small value, then this suggests that we have not done a good job, a priori, of predicting our data, suggesting remodeling of our likelihood or prior or both. Or, similarly, we can compare observed data to fake data generated from $[\mathbf{y}]$ in, e.g., a **plot of observed vs. (prior) predicted values**.

- Also, the prior predictive distribution is sometimes used in an **empirical Bayes** analysis to specify a prior distribution; sometimes called **type-II maximum likelihood** ([Ber85]). While we have integrated/summed over $\boldsymbol{\theta}$ to obtain $[\mathbf{y}]$, $[\mathbf{y}]$ still contains the ("hyper") parameters of the prior $[\boldsymbol{\theta}]$, and these need to be given particular values. Thus, we view $[\mathbf{y}]$ as a likelihood to be maximized, the resulting estimates of the hyperparameters plugged into $[\boldsymbol{\theta}]$, which is then used in a subsequent analysis. Incidentally, **machine learners** refer to this approach to prior specification as **evidence approximation** ([Bis06, §3.5]). Empirical Bayes is **not strictly Bayesian** because we use the data to determine parameters of the prior distribution, which, as the name suggests, should be determined before looking at the data.

More forwardly, of course, we may want to make predictions about unobserved outputs using observations that we have observed.

- This goal is embodied (in principle) in the **(posterior) predictive distribution**, $[\mathbf{y}^* \,|\, \mathbf{y}]$.

- As mentioned, in principle, we can obtain this by integrating/summing over the joint posterior distribution $[\mathbf{y}^*, \boldsymbol{\theta} \,|\, \mathbf{y}]$, which we may factor (by the multiplication rule, Definition 5.6 INF511Notes.pdf) as

$$[\mathbf{y}^*, \boldsymbol{\theta} \,|\, \mathbf{y}] = [\mathbf{y}^* \,|\, \mathbf{y}, \boldsymbol{\theta}][\boldsymbol{\theta} \,|\, \mathbf{y}].$$

- Thus, in principle,

$$[\mathbf{y}^* \,|\, \mathbf{y}] = \int [\mathbf{y}^* \,|\, \mathbf{y}, \boldsymbol{\theta}][\boldsymbol{\theta} \,|\, \mathbf{y}] \mathrm{d}\boldsymbol{\theta},$$

  where $[\mathbf{y}^* \,|\, \mathbf{y}, \boldsymbol{\theta}]$ is the distribution of unobserved data (unobserved response/outputs) $\mathbf{y}^*$ conditional on (observed) $\mathbf{y}$, which we can write as

$$[\mathbf{y}^* \,|\, \mathbf{y}, \boldsymbol{\theta}] = [\mathbf{y}^* \,|\, \boldsymbol{\theta}],$$

  if $\mathbf{y}^*$ and $\mathbf{y}$ are independent; as in, e.g., the classical linear model. See [Wak13, Expr. (3.9)].

- As previously discussed in the case of the posterior, $[\boldsymbol{\theta} \,|\, \mathbf{y}]$, **if we recognize the kernel of the posterior predictive**, great; we have our answer (up to summarization anyway). ("Hey, that's a t distribution!")

- But, generally, as with $[\boldsymbol{\theta} \,|\, \mathbf{y}]$, for all but the simplest models, **we may not recognize the kernel of the posterior predictive**, and we may think that we have to evaluate the integral. But, using MCMC procedures, practically speaking, we do not have to evaluate the integral (well, technically, we more/less evaluate it via MCMC integration, but this is sort of hidden, as mentioned).

- Again, **linear models are relatively simple** in that we often recognize (conditional) kernels, but, still, we may use (MC)MC anyway as a convenient way to get posterior summaries: mean, variance, intervals, etc.

- Notice, because $[\mathbf{y}^* \,|\, \mathbf{y}, \boldsymbol{\theta}]$ (or, with independence, $[\mathbf{y}^* \,|\, \boldsymbol{\theta}]$) typically follows from the specification of the data model (e.g., regression model), this gives us a relatively easy way to obtain (samples from) the posterior predictive. That is, if we can obtain samples of $\boldsymbol{\theta}$ from the (marginal conditional!) posterior $[\boldsymbol{\theta} \,|\, \mathbf{y}]$ (from MCMC for example), and if we know the conditional $[\mathbf{y}^* \,|\, \mathbf{y}, \boldsymbol{\theta}]$ (again, it should be obvious from our data model!), then we can generate $\mathbf{y}^* \,|\, \boldsymbol{\theta}, \mathbf{y}$ so that we obtain samples of $(\mathbf{y}^*, \boldsymbol{\theta} \,|\, \mathbf{y})$ from the joint posterior...one $\boldsymbol{\theta} \,|\, \mathbf{y}$ then one $\mathbf{y}^* \,|\, \boldsymbol{\theta}, \mathbf{y}$, another $\boldsymbol{\theta} \,|\, \mathbf{y}$ then another $\mathbf{y}^* \,|\, \boldsymbol{\theta}, \mathbf{y}$, etc. If you want a sample from the (marginal posterior), $[\mathbf{y}^* \,|\, \mathbf{y}]$ (our posterior predictive), then just ignore the sample of $\boldsymbol{\theta} \,|\, \mathbf{y}$ values (and vice-versa)! This process sometimes called **composition sampling**; see [Wak13, §3.8.4]

4.4 Joint Posterior

Incidentally, we often see the posterior, $[\boldsymbol{\theta} \,|\, \mathbf{y}]$ and the posterior predictive, $[\mathbf{y}^* \,|\, \mathbf{y}]$, discussed somewhat separately. As mentioned, we can simply view our goal as the **joint posterior distribution** (just Bayes theorem again),

$$\begin{aligned}
[\mathbf{y}^*, \boldsymbol{\theta} \,|\, \mathbf{y}] &= [\mathbf{y}^* \,|\, \mathbf{y}, \boldsymbol{\theta}][\boldsymbol{\theta} \,|\, \mathbf{y}] \\
&= [\mathbf{y}^* \,|\, \mathbf{y}, \boldsymbol{\theta}] \frac{[\mathbf{y} \,|\, \boldsymbol{\theta}][\boldsymbol{\theta}]}{[\mathbf{y}]} \\
&\propto [\mathbf{y}^* \,|\, \mathbf{y}, \boldsymbol{\theta}][\mathbf{y} \,|\, \boldsymbol{\theta}][\boldsymbol{\theta}] \\
&= [\mathbf{y}^*, \mathbf{y}, \boldsymbol{\theta}]
\end{aligned}$$

again seeing the numerator of Bayes theorem as a **joint distribution constructed hierarchically** using the multiplication rule (Definition 5.6 INF511Notes.pdf).

Thus, given methods to implement Bayes theorem, the practical challenge is to specify a joint probability model for our (un)observed data and parameters. Thus, we have probabilistic programming languages, e.g., WinBUGS, OpenBUGS, JAGS, Stan and NIMBLE, that facilitate the specification of such joint models, with implementation details largely behind the scenes.

4.5 Summary I

To summarize/reiterate, we have (i.e., we specify) the ((un)observed) data distribution (likelihood), $[\mathbf{y} \,|\, \boldsymbol{\theta}]$ or $[\mathbf{y}^*, \mathbf{y} \,|\, \boldsymbol{\theta}]$ and the prior, $[\boldsymbol{\theta}]$which, in principle, give the (joint) posterior, $[\boldsymbol{\theta} \,|\, \mathbf{y}]$ or $[\mathbf{y}^*, \boldsymbol{\theta} \,|\, \mathbf{y}]$ via Bayes theorem, up to a normalizing constant anyway, and, after "integrating" the joint posterior, we may focus on the posterior predictive, $[\mathbf{y}^* \,|\, \mathbf{y}]$.

Given our discussion of the **kernel** as the "important part" of a ((joint) posterior) distribution, we might think we are done if we recognize the kernel, because, in this case, we can avoid having to compute the integral/sum to get the normalizing constant $[\mathbf{y}]$ (prior predictive). But, again, this requires that we recognize the form of the kernel to correspond to a known distribution, perhaps, e.g., normal, normal-gamma or t, from which we could compute easier-to-grasp summarizing quantities for our posterior, like means, medians, modes, quantiles, intervals, etc., using ordinary methods implemented in many softwares.

When we fail to recognize kernels, we avail ourselves of other methods, such as MCMC, which usually does not require the kernel.

4.6 Now in Terms of the Linear Model

- Recall our normal linear model from Chapter 2 of our lecture notes,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma}),$$

  where, for much of what we will do, $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$.

- Thus, following our generic Bayesian discussion, above, we write our **data distribution** in the current context as

$$[\mathbf{y} \,|\, \boldsymbol{\beta}, \sigma^2].$$

- We may also state the linear model for an unobserved output vector,

$$\mathbf{y}^* = \mathbf{X}^*\boldsymbol{\beta} + \boldsymbol{\epsilon}^*, \quad \boldsymbol{\epsilon}^* \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma}),$$

with corresponding notation

$$[\mathbf{y}^* \,|\, \boldsymbol{\beta}, \sigma^2]$$

- And, we may write the joint (un)observed data distribution as

$$\begin{aligned}[\mathbf{y}^*, \mathbf{y} \,|\, \boldsymbol{\beta}, \sigma^2] &= [\mathbf{y}^* \,|\, \mathbf{y}, \boldsymbol{\beta}, \sigma^2][\mathbf{y} \,|\, \boldsymbol{\beta}, \sigma^2] \\ &= [\mathbf{y}^* \,|\, \boldsymbol{\beta}, \sigma^2][\mathbf{y} \,|\, \boldsymbol{\beta}, \sigma^2],\end{aligned}$$

the second equality following from the (conditional) independence of outputs.

- In the classic Bayesian linear model, as in the frequentist version, $\mathbf{x}$ (hence $\mathbf{X}$) will be considered known, and we do not consider the distribution of covariate/input $\mathbf{x}$ (see §5.3, INF511Notes.pdf). This assumes a factorization of the prior analogous to that for $[y, \mathbf{x}]$ so that we do not have to consider the prior for the parameters of the covariate/input distribution, either. We skip further discussion of this. (Same for $\mathbf{x}^*$ and $\mathbf{X}^*$ of course.)

- We may not use notation to explicitly indicate conditioning on $\mathbf{x}$ or $\mathbf{X}$ (or starred versions).

- Continuing to follow the above, generic Bayesian development, we must specify a **prior distribution** for the parameters, which, in the current context, we denote as

$$[\boldsymbol{\beta}, \sigma^2].$$

- We will discuss common prior distributional specifications and corresponding (joint) **posterior distribution**, which, in our current context, we denote as

$$[\boldsymbol{\beta}, \sigma^2 \,|\, \mathbf{y}],$$

or

$$[\mathbf{y}^*\boldsymbol{\beta}, \sigma^2 \,|\, \mathbf{y}].$$

- And, we well will discuss the corresponding marginal **posterior predictive** distribution, which we denote now as

$$[\mathbf{y}^* \,|\, \mathbf{y}].$$

- Let's jump into common prior distributions, $[\boldsymbol{\beta}, \sigma^2]$, for the linear model.

4.7 Conjugate Prior

- The **conjugate prior distribution** for the mean parameter, $\boldsymbol{\beta}$, and variance parameter, σ^2, considered collectively in the normal linear model, is the product (using the multiplication rule) of a **conditional normal distribution** and a **scaled inverse-chi-square**,

$$\begin{aligned}[\boldsymbol{\beta}, \sigma^2] &= [\boldsymbol{\beta} \,|\, \sigma^2][\sigma^2] \\ &= \mathrm{N}(\boldsymbol{m}_0, \sigma^2 \boldsymbol{\Sigma}_0) \times \text{inv-}\chi^2(\nu_0, \sigma_0^2)\end{aligned}$$

  where we must specify values (or further distributions—not here) for the prior (hyper)parameters,

$$\begin{aligned}\boldsymbol{m}_0 &= \text{conditional prior } \textbf{mean}, \\ \sigma^2 \boldsymbol{\Sigma}_0 &= \text{conditional prior } \textbf{variance}, \\ \nu_0 &= \text{marginal prior } \textbf{degrees of freedom} \\ \sigma_0^2 &= \text{marginal prior } \textbf{scale} \text{ (squared)}.\end{aligned}$$

- See [Wak13, §3.7.1 & p. 223] for a brief discussion of conjugacy. (Incidentally, inv-$\chi^2(\nu_0, \sigma_0^2)$ is the same as inv-gamma$(a = \nu_0/2, b = \sigma_0^2 \nu_0/2)$.)
- Shortly, we will see more intuition for why ν_0 and σ_0^2 are called degrees of freedom and scale (squared), respectively, and hence why $\nu_0 \sigma_0^2$, may be seen as a **prior sum-of-squares**.

- We will attempt to follow the **distribution parameterizations** of [GCS+14, Appendix A], unless otherwise indicated.
- A **conjugate prior distribution** for the parameter of a data distribution is such that the posterior distribution of that parameter is of the **same distributional form** as the prior distribution for that parameter ([Rob01, Def. 3.3.1]). The data distribution and the conjugate prior are often said to form a **conjugate pair** of distributions.
- Thus, given our conjugate prior above, e.g., we should expect the posterior (see below) to also be the product of a **conditional normal distribution** and a **scaled inverse-chi-square** (with different hyperparameters that are somehow updated to incorporate our observed data, of course).
- Conjugacy is not only convenient, but has **theoretical justification**. For certain families of data distributions, including ours, here, mixtures of conjugate distributions are also conjugate, and we can get arbitrarily close to many distributions with such mixtures ([Rob01, Lemma 3.4.2 & Theorem 3.4.3]). Thus, in principle, mixtures provide flexibility in specifying priors.

4.7.1 Posterior

- According to **Bayes theorem**, we have

$$[\boldsymbol{\beta}, \sigma^2 \,|\, \mathbf{y}] \;\propto\; [\mathbf{y} \,|\, \boldsymbol{\beta}, \sigma^2][\boldsymbol{\beta} \,|\, \sigma^2][\sigma^2],$$

  which, by conjugacy (and other standard results), can be shown to give the **posterior**

$$\begin{aligned} [\boldsymbol{\beta}, \sigma^2 \,|\, \mathbf{y}] &= [\boldsymbol{\beta} \,|\, \sigma^2, \mathbf{y}][\sigma^2 \,|\, \mathbf{y}] \\ &= \mathrm{N}(\widehat{\boldsymbol{m}}, \sigma^2 \widehat{\boldsymbol{\Sigma}}) \times \text{inv-}\chi^2(\widehat{\nu}, \widehat{\sigma}^2) \end{aligned}$$

where

$$\begin{aligned}
\widehat{\boldsymbol{m}} &= (\boldsymbol{I}-\boldsymbol{W})\boldsymbol{m}_0 + \boldsymbol{W}\widehat{\boldsymbol{\beta}} \quad \text{cond. post. } \textbf{mean}, \\
\sigma^2\widehat{\boldsymbol{\Sigma}} &= \boldsymbol{W}\mathrm{Var}(\widehat{\boldsymbol{\beta}}) \quad \text{cond. post. } \textbf{variance}, \\
\widehat{\nu} &= \nu_0 + n \quad \text{post. } \textbf{df} \\
\widehat{\sigma}^2 &= (1/\widehat{\nu})(\nu_0\sigma_0^2 + (\mathbf{y}-\mathbf{X}\widehat{\boldsymbol{\beta}})^t(\mathbf{y}-\mathbf{X}\widehat{\boldsymbol{\beta}}) + \\
&\quad (\boldsymbol{m}_0-\widehat{\boldsymbol{\beta}})^t(\boldsymbol{\Sigma}_0+(\mathbf{X}^t\mathbf{X})^{-1})^{-1}(\boldsymbol{m}_0-\widehat{\boldsymbol{\beta}}))\text{post. } \textbf{scale} \text{ (squared)} \\
\boldsymbol{W} &= (\sigma^{-2}\boldsymbol{\Sigma}_0^{-1}+\sigma^{-2}\mathbf{X}^t\mathbf{X})^{-1}\sigma^{-2}\mathbf{X}^t\mathbf{X} \\
&= (\sigma^{-2}\boldsymbol{\Sigma}_0^{-1}+\mathrm{Var}(\widehat{\boldsymbol{\beta}})^{-1})^{-1}\mathrm{Var}(\widehat{\boldsymbol{\beta}})^{-1}
\end{aligned}$$

- Thus, conjugacy simply requires that we use (well-established) **update formulas** to get from our prior to our posterior; the result of Bayes theorem has largely been obtained for us with relatively little effort (just some algebra, which you will undoubtedly find all over the Web).

- In other words, we do not need to find the normalizing constant or to appeal to methods such as MCMC, for samples from the posterior, or INLA, as a numerical solution to the posterior. Still, we may want to summarize the posterior in some way, so, we may obtain, e.g., ordinary Monte Carlo samples from the posterior using, e.g., `rt`, for $\boldsymbol{\beta}$.

- In other words, someone recognized that the form of the prior, when compared to the data model, would result in a posterior whose kernel is proportional to a normal-inv-χ^2, and merely requiring some algebra to find the typical (posterior) parameters without having to integrate to get the normalizing constant.

- **Notice a few things**:

  - The conditional normal **posterior mean** of $\boldsymbol{\beta}$ is a weighted average of the prior mean of $\boldsymbol{\beta}$ and a data-based (unbiased) estimate of $\boldsymbol{\beta}$.

- The conditional normal **posterior variance** of $\boldsymbol{\beta}$ is the inverse of the sum of its prior precision (inverse of variance) and the precision of a data-based estimate, which may be easier to see after a bit of algebra.
- The marginal **posterior df** is a sum of the prior degrees of freedom and sample size.
- The marginal **posterior sum-of-squares** is a sum of the prior sum-of-squares (hence we see suggestive name), the usual error sum-of-squares (SSE) and a sum-of-squares that is a sort of discrepancy between the prior and data-based means of $\boldsymbol{\beta}$.
- **As the (conditional) prior variance "increases,"** the discrepancy between prior (conditional) mean and data-based mean is down-weighted, making the posterior scale (squared) smaller hence making the posterior variance (and mean) of σ^2 smaller. (You need know the mean and variance of a scaled inv-χ^2 for this comment to make sense ([GCS+14, Appendix A]).)
- **As prior df increases** the posterior scale (squared) looks more like the prior scale (squared) hence the posterior mean (and variance) of σ^2 looks more like the prior scale (squared). (You need know the mean and variance of a scaled inv-χ^2 for this to make sense ([GCS+14, Appendix A]).)

4.7.2 Marginal Posterior for $\boldsymbol{\beta}$ is a t

- The **marginal posterior**, $[\boldsymbol{\beta} \,|\, \mathbf{y}]$, is what we would use to infer about $\boldsymbol{\beta}$. In some (rough) sense, it's the Bayesian counterpart to the normal/χ^2/t/F results in Chapter 6, INF511Notes.pdf, and summarized in Chapter 2, here, for inferring about $\boldsymbol{\beta}$. (But, it's somehow much different!)
- We will use the following **result about a t distribution** to get $[\boldsymbol{\beta} \,|\, \mathbf{y}]$

(and to get the posterior predictive, $[\mathbf{y}^* \,|\, \mathbf{y}]$, shortly thereafter).

- The (generic, not sample size) n dimensional t distribution (pdf),

$$t_n(\nu_0, \mathbf{m}_0, \sigma_0^2 \mathbf{\Sigma}_0),$$

is often defined as a **scale mixture** of a conditional normal with an inv-χ^2 (scaled inverse chi-square) mixing distribution (alternatively and equivalently, an inv-gamma mixing distribution, but I prefer the parameterization of an inv-χ^2).

- In other words, the t pdf is often defined as the **marginal** distribution,

$$[\boldsymbol{\theta}] = t_n(\nu_0, \mathbf{m}_0, \sigma_0^2 \mathbf{\Sigma}_0),$$

where $(\boldsymbol{\theta}^t, \sigma^2)^t$ has **joint** distribution

$$[\boldsymbol{\theta}, \sigma^2] = [\boldsymbol{\theta}|\sigma^2][\sigma^2]$$

defined by the **conditional** normal

$$\boldsymbol{\theta}|\sigma^2 \sim N_n(\mathbf{m}_0, \sigma^2 \mathbf{\Sigma}_0)$$

and **marginal** inv-χ^2

$$\sigma^2 \sim \text{inv-}\chi^2(\nu_0, \sigma_0^2).$$

- This means, going the other way, any time we see this form for a joint distribution, we can immediately write down the marginal

$$[\boldsymbol{\theta}] = t_n(\nu_0, m_0, \sigma_0^2 \mathbf{\Sigma}_0).$$

- Incidentally, this is multivariate t, that, appropriately centered and scaled, would give univariate t's of "STAT 101," which we know as **Student's** t. (To be sure, t_n is ***not*** what is known as a non-central t.)

- In other words, if you multiplied a standard (Student's) t (with $df = \nu_0$) by a scalar, σ_0, then added m_0, you would have a variable distributed as
$$t \sim t_1(\nu_0, m_0, \sigma_0^2),$$
where $\mathrm{E}(t) = m_0$ ($\nu_0 > 1$) and $\mathrm{Var}(t) = \frac{\nu_0}{\nu_0 - 2}\sigma_0^2$ ($\nu_0 > 2$).
- (NOTE: [GCS+14, Appendix A] absorbs σ_0^2 into $\boldsymbol{\Sigma}_0$, and uses different symbols as does [Wak13, Appendix D], in a different order.)
- Thus, with this mixture result and suggestive use of prior hyperparameter symbols, it should be obvious that the **marginal prior** for $\boldsymbol{\beta}$ is
$$[\boldsymbol{\beta}] = \mathrm{t}_p(\nu_0, \boldsymbol{m}_0, \sigma_0^2\boldsymbol{\Sigma}_0)$$
- Of course, we likely want the **marginal (t) posterior**, which follows immediately with an obvious change in symbols,
$$[\boldsymbol{\beta} \,|\, \mathbf{y}] = \mathrm{t}_p(\widehat{\nu}, \widehat{\boldsymbol{m}}, \widehat{\sigma}^2\widehat{\boldsymbol{\Sigma}}),$$
where all updated posterior parameters are as defined above.

4.7.3 Posterior Predictive is a t

- What's the **posterior predictive**, $[\mathbf{y}^* \,|\, \mathbf{y}]$?
- Our model for n^* unobserved, "future" responses/outputs, which we denote as $\mathbf{y}^*$, to distinguish them from observed inputs, $\mathbf{y}$, follow our same data model,
$$\mathbf{y}^* \sim \mathrm{N}(\mathbf{X}^*\boldsymbol{\beta}, \sigma^2\boldsymbol{I}),$$
independent of $\mathbf{y}$.
- Thus, we have a **joint data (unobserved and observed) distribution**
$$[\mathbf{y}^*, \mathbf{y} \,|\, \boldsymbol{\beta}, \sigma^2] = [\mathbf{y}^* \,|\, \mathbf{y}, \boldsymbol{\beta}, \sigma^2][\mathbf{y} \,|\, \boldsymbol{\beta}, \sigma^2] = [\mathbf{y}^* \,|\, \boldsymbol{\beta}, \sigma^2][\mathbf{y} \,|\, \boldsymbol{\beta}, \sigma^2],$$

which we can use, along with the prior, to get our joint probability model for all quantities, hence to get the (joint) posterior,

$$\begin{aligned}
[\mathbf{y}^*, \boldsymbol{\beta}, \sigma^2 \,|\, \mathbf{y}] &\propto [\mathbf{y}^*, \mathbf{y}, \boldsymbol{\beta}, \sigma^2] \\
&= [\mathbf{y}^* \,|\, \boldsymbol{\beta}, \sigma^2][\mathbf{y} \,|\, \boldsymbol{\beta}, \sigma^2][\boldsymbol{\beta} \,|\, \sigma^2][\sigma^2] \\
&\propto [\mathbf{y}^* \,|\, \boldsymbol{\beta}, \sigma^2] \frac{[\mathbf{y} \,|\, \boldsymbol{\beta}, \sigma^2][\boldsymbol{\beta} \,|\, \sigma^2][\sigma^2]}{[\mathbf{y}]} \\
&= [\mathbf{y}^* \,|\, \boldsymbol{\beta}, \sigma^2][\boldsymbol{\beta}, \sigma^2 \,|\, \mathbf{y}] \\
&= [\mathbf{y}^* \,|\, \boldsymbol{\beta}, \sigma^2][\boldsymbol{\beta} \,|\, \sigma^2, \mathbf{y}][\sigma^2 \,|\, \mathbf{y}] \\
&= [\mathbf{y}^*, \boldsymbol{\beta} \,|\, \sigma^2, \mathbf{y}][\sigma^2 \,|\, \mathbf{y}],
\end{aligned}$$

where $[\mathbf{y}^*, \boldsymbol{\beta} \,|\, \sigma^2, \mathbf{y}]$ is a **conditional normal** distribution (a joint distribution of $\mathbf{y}^*$ and $\boldsymbol{\beta}$ conditional on σ^2 and $\mathbf{y}$) built up hierarchically from the conditional normal posterior $[\boldsymbol{\beta} \,|\, \sigma^2, \mathbf{y}]$ and the normal unobserved data model $[\boldsymbol{y}^* \,|\, \boldsymbol{\beta}, \sigma^2] = [\boldsymbol{y}^* \,|\, \mathbf{y}, \boldsymbol{\beta}, \sigma^2]$, and

$$[\sigma^2 \,|\, \mathbf{y}]$$

is the same **marginal inv-χ^2** posterior, above.

- $\mathbf{y}^*$ and $\boldsymbol{\beta}$ collectively now play the role of $\boldsymbol{\beta}$, or of the generic $\boldsymbol{\theta}$, in the above t (mixture) result. Thus, we could apply the above t result, "integrating out" σ^2, to get a joint t for $[\mathbf{y}^*, \boldsymbol{\beta} \,|\, \mathbf{y}]$.

- But, we are now interested in $\mathbf{y}^*$, in particular, not both $\mathbf{y}^*$ and $\boldsymbol{\beta}$, and we already know that the marginal posterior for $[\boldsymbol{\beta} \,|\, \mathbf{y}]$ is a t, so we first "integrate out" $\boldsymbol{\beta}$ (not!), to give a marginal posterior that is the **product of a conditional normal posterior predictive and the same inv-χ^2**,

$$[\mathbf{y}^*, \sigma^2 \,|\, \mathbf{y}] = [\mathbf{y}^* \,|\, \sigma^2, \mathbf{y}][\sigma^2 \,|\, \mathbf{y}],$$

where, in particular, (skipping some details)

$$\begin{aligned}
[\mathbf{y}^* \,|\, \sigma^2, \mathbf{y}] &= \mathrm{N}(\widehat{\boldsymbol{m}}^*, \sigma^2 \widehat{\boldsymbol{\Sigma}}^*), \\
\widehat{\boldsymbol{m}}^* &= \mathbf{X}^* (\boldsymbol{\Sigma}_0^{-1} + (\mathbf{X}^t \mathbf{X}))^{-1} (\boldsymbol{\Sigma}_0^{-1} \boldsymbol{m}_0 + (\mathbf{X}^t \mathbf{X}) \widehat{\boldsymbol{\beta}}), \\
&= \mathbf{X}^* \widehat{\boldsymbol{m}} \\
\widehat{\boldsymbol{\Sigma}}^* &= (\boldsymbol{I} + \mathbf{X}^* (\boldsymbol{\Sigma}_0^{-1} + (\mathbf{X}^t \mathbf{X}))^{-1} \mathbf{X}^{*t}), \\
&= (\boldsymbol{I} + \mathbf{X}^* \widehat{\boldsymbol{\Sigma}} \mathbf{X}^{*t}),
\end{aligned}$$

and, again, $[\sigma^2 \,|\, \mathbf{y}]$ is the same inv-χ^2 as above with the same df and scale parameters as defined above.

- Thus, we can again use the above t result to write down the desired **posterior predictive** distribution as

$$[\mathbf{y}^* \,|\, \mathbf{y}] = t_{n^*}(\widehat{\nu}, \widehat{\boldsymbol{m}}^*, \widehat{\sigma}^2 \widehat{\boldsymbol{\Sigma}}^*),$$

where all posterior parameters have been defined above.

4.7.4 Remarks

- Alas, the above **conjugate prior** (normal×inv-χ^2) for linear models may seem **unrealistic** in the sense that $\boldsymbol{\beta}$ becomes increasingly concentrated (disperse) around its prior mean, $\boldsymbol{m}_0$, as the data model error variance, σ^2, becomes smaller (larger). Or, this just may otherwise seem like an unnatural way to express prior information about $\boldsymbol{\beta}$.
- **Zellner's G-prior** fits into the current discussion (and that of the next section, §4.8) by specifying the conditional normal prior variance matrix as

$$\sigma^2 \boldsymbol{\Sigma}_0 = \sigma^2 g (\mathbf{X}^t \mathbf{X})^{-1},$$

where $g > 0$ is given a value or a prior (and we set the prior $[\sigma^2] \propto \sigma^{-2}$...§4.8, below), which seems to alleviate the previous item's

concern, plus we seem to have a more informative prior on $\boldsymbol{\beta}$ in the sense of using its (known covariate/input) data-based covariance structure in $(\mathbf{X}^t\mathbf{X})^{-1}$). Zellner's G-prior is traditionally used in **Bayesian variable selection**, which we treat in a subsequent chapter.

- Or, we may consider **other priors**. But, generally speaking, we do not get (full) conjugacy, but, instead, conditional conjugacy or perhaps no conjugacy (unusual for typical normal linear models).
- In the next section, we consider a popular **non-informative**, **improper prior**, that results in effectively the same form of posterior and posterior predictive (and our work above is not for naught!).

4.8 A Common Improper Prior

- A common **improper prior** distribution for the normal linear model is ([Wak13, Expr. (5.42)])

$$[\boldsymbol{\beta}, \sigma^2] \propto \sigma^{-2}.$$

- Incidentally, this is the product of **Jeffreys' prior** for $\boldsymbol{\beta}$ ($\propto 1$) and **Jeffreys' prior** for σ^2 (but, perhaps confusingly, is *not* Jeffreys' prior for (jointly) $(\boldsymbol{\beta}, \sigma^2)$!...)
- An **improper prior** is one that does not integrate (or sum) to a finite value, thus it cannot be normalized to integrate (or sum) to 1.
- It's the **propriety of the posterior** that counts.
- Loosely, this improper prior can be viewed as the previous section's proper conjugate prior with marginal prior df $\nu_0 = 0$ and conditional prior precision $(\sigma^2\boldsymbol{\Sigma}_0)^{-1} = \mathbf{0}$. (If you plugged these values into the conjugate pdf, you'd see what I mean.)

- It's **not a conjugate prior** (unless we consider the improper case as an extended member of the normal-inv-χ^2 conjugate family...).

4.8.1 Posterior

- In short (skipping details), we have the posterior

$$\begin{aligned}[\boldsymbol{\beta}, \sigma^2 \,|\, \mathbf{y}] &= [\boldsymbol{\beta} \,|\, \sigma^2, \mathbf{y}][\sigma^2 \,|\, \mathbf{y}] \\ &= \mathrm{N}(\widehat{\boldsymbol{m}}, \sigma^2 \widehat{\boldsymbol{\Sigma}}) \times \mathrm{inv}-\chi^2(\widehat{\nu}, \widehat{\sigma}^2)\end{aligned}$$

  where

$$\begin{aligned}\widehat{\boldsymbol{m}} &= (\boldsymbol{I} - \boldsymbol{W})\boldsymbol{m}_0 + \boldsymbol{W}\widehat{\boldsymbol{\beta}} \quad \text{cond. post. } \textbf{mean}, \\ &= \widehat{\boldsymbol{\beta}} \\ \sigma^2 \widehat{\boldsymbol{\Sigma}} &= \boldsymbol{W}\mathrm{Var}(\widehat{\boldsymbol{\beta}}) \quad \text{cond. post. } \textbf{variance}, \\ &= \mathrm{Var}(\widehat{\boldsymbol{\beta}}) \\ \widehat{\nu} &= n - p \quad \text{post. } \textbf{df} \\ \widehat{\sigma}^2 &= (1/\widehat{\nu})(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})^t(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}) \quad \text{post. } \textbf{scale} \text{ squared} \\ \boldsymbol{W} &= (\mathbf{0} + \mathrm{Var}(\widehat{\boldsymbol{\beta}})^{-1})^{-1}\mathrm{Var}(\widehat{\boldsymbol{\beta}})^{-1} \\ &= \boldsymbol{I}\end{aligned}$$

- NOTE: Now, in this improper prior case, we must have $n > p$ and $\mathbf{X}$ must be full rank, neither of which were strictly necessary in the conjugate case, above, as long as $\sigma^2\boldsymbol{\Sigma}_0$ was a valid (positive definite) variance matrix in that case.
- Things are looking strangely familiar...

4.8.2 Marginal Posterior for $\boldsymbol{\beta}$ is a Familiar t

- Again, the posterior (however, not the prior, now!) is of the form of the above t scale mixture result, giving **marginal posterior**

$$[\boldsymbol{\beta} \,|\, \mathbf{y}] = \mathrm{t}_p(n-p, \widehat{\boldsymbol{\beta}}, \widehat{\sigma}^2(\mathbf{X}^t\mathbf{X})^{-1})$$

  ([Wak13, Expr. 5.44], $p = k+1$, and his parameter arguments are in a different order).

- In particular, the MLE (and LS estimator) is the same as the (marginal) posterior mean, $\widehat{\boldsymbol{\beta}}$ $(n-p>1)$, and the posterior variance is the same as $\widehat{Var}(\widehat{\boldsymbol{\beta}})$ $(n-p>2)$!

- In other words, each individual posterior marginal random variable, β_j is

$$\mathrm{t}_1(n-p, \widehat{\beta}_j, \widehat{\sigma}^2(\mathbf{X}^t\mathbf{X})^{-1}_{(jj)})$$

  where $(\mathbf{X}^t\mathbf{X})^{-1}_{(jj)}$ is the jth diagonal element of $(\mathbf{X}^t\mathbf{X})^{-1}$.

- In other words, standardizing, we get

$$\frac{\beta_j - \widehat{\beta}_j}{\sqrt{\widehat{\sigma}^2(\mathbf{X}^t\mathbf{X})^{-1}_{jj}}} \sim t(n-p) \quad whaoahuh!?$$

- This should look **very familiar** (see Result 6.10, Chapter 6, INF511Notes.pdf, which we restated, briefly, in Chapter 2 here (using a certain $\mathbf{C}$ row matrix). But, in some sense, this result is **very different** as, now, in the Bayesian context, the randomness comes from $\boldsymbol{\beta}$ and not from $\widehat{\boldsymbol{\beta}}$ and $\widehat{\sigma}^2$! (See related comment middle of [Wak13, p. 222].)

- To be sure, $\boldsymbol{\beta}$ (β_j) is now random, not $\widehat{\boldsymbol{\beta}}$ $(\widehat{\beta}_j)$, which is now fixed, as is $\widehat{\sigma}^2$.

- (In Chapter 6, INF511Notes.pdf, we could have stated a multi-variate t result for $\widehat{\boldsymbol{\beta}}$, but we didn't.)

4.8.3 Posterior Predictive is a Familiar t

- And, in short, we have the **posterior predictive**

$$[\mathbf{y}^* \,|\, \mathbf{y}] = t_{n^*}(\widehat{\nu}, \widehat{\boldsymbol{m}}^*, \widehat{\sigma}^2 \widehat{\Sigma}^*),$$

  where (we've skipped a few details)

$$\begin{aligned} \widehat{\boldsymbol{m}}^* &= \mathbf{X}^* \widehat{\boldsymbol{\beta}}, & (4.1)\\ \widehat{\boldsymbol{\Sigma}}^* &= (\boldsymbol{I} + \mathbf{X}^* (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^{*t}), & (4.2)\\ \widehat{\nu} &= n - p \quad , & (4.3)\\ \widehat{\sigma}^2 &= (1/\widehat{\nu})(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})^t (\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}) & (4.4) \end{aligned}$$

- In other words,

$$[\mathbf{y}^* \,|\, \mathbf{y}] = t_{n^*}(n - p, \mathbf{X}^* \widehat{\boldsymbol{\beta}}, \widehat{\sigma}^2 (\boldsymbol{I} + \mathbf{X}^* (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^{*t})),$$

  which means $y_i^* \,|\, \mathbf{y}$ is

$$t_1(n - p, \mathbf{x}_i^{*t} \widehat{\boldsymbol{\beta}}, \widehat{\sigma}^2 (1 + \mathbf{x}^{*t} (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{x}^*)).$$

- In other words, if we standardize, we get

$$\frac{y_i^* - \mathbf{x}_i^{*t} \widehat{\boldsymbol{\beta}}}{\sqrt{\widehat{\sigma}^2 (1 + \mathbf{x}_i^{*t} (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{x}_i^*)}} \sim t(n - p)$$

- This, too, should look very familiar, but, while y_i^* is random, Bayesian or not, $\widehat{\boldsymbol{\beta}}$ and $\widehat{\sigma}^2$ are now, again, fixed.

4.9 STAT 101 Redux a la Bayes

See INF511Notes.pdf §8.7 for an illustration of the previous section's **particular improper prior** and connection to typical frequentist linear model results.

4.10 A Common Independence Prior

$$\begin{aligned}[\boldsymbol{\beta}, \sigma^2] &= [\boldsymbol{\beta}][\sigma^2] \\ &= \mathrm{N}(\boldsymbol{m}_0, \boldsymbol{\Sigma}_0) \times \text{inv-}\chi^2(\nu_0, \sigma_0^2)\end{aligned}$$

- Notice, in particular, that we now assume **independence** a priori. See middle of [Wak13, p. 223]. (Again, an inverse gamma is equivalent to an inv-χ^2, and saying $\sigma^{-2} \sim \text{gamma}(\alpha, \beta)$ is the same as saying $\sigma^2 \sim \text{inv-gamma}(\alpha, \beta)$.)
- Now, with this independence prior, **we do not get a closed form posterior**, like those resulting from the two priors considered previously, unless we consider let $\boldsymbol{\Sigma}_0^{-1} = \mathbf{0}$ and $\nu_0 = 0$, which gives the previous improper prior and closed-form $\mathrm{N} \times \text{inv-}\chi^2$ posterior results.
- It's **not a conjugate prior** (unless we consider the improper case as member of the normal-inv-χ^2 family...).
- In other words, we do not get a recognizable posterior distribution (for which we usually know means, variances, probabilities, `R` functions, etc.).
- Now what? (wait a moment)

4.10.1 Full Conditional Posterior Distributions

We do however know the form of **full conditional posterior distributions** (often shortened to "**full conditionals**")

$$\begin{aligned}\boldsymbol{\beta} \,|\, \sigma^2, \mathbf{y} &\sim \mathrm{N}(\widehat{\boldsymbol{m}}, \sigma^2 \widehat{\boldsymbol{\Sigma}}) \\ \sigma^2 \,|\, \boldsymbol{\beta}, \mathbf{y} &\sim \text{inv-}\chi^2(\widehat{\nu}, \widehat{\sigma}^2),\end{aligned}$$

where

$$\begin{aligned}
\widehat{\boldsymbol{m}} &= (\boldsymbol{I}-\boldsymbol{W})\boldsymbol{m}_0 + \boldsymbol{W}\widehat{\boldsymbol{\beta}} \quad \text{cond. post. } \textbf{mean}, \\
\sigma^2\widehat{\boldsymbol{\Sigma}} &= \boldsymbol{W}\mathrm{Var}(\widehat{\boldsymbol{\beta}}) \quad \text{cond. post. } \textbf{variance}, \\
\widehat{\nu} &= \nu_0 + n \quad \text{post. } \textbf{df} \\
\widehat{\sigma}^2 &= (1/\widehat{\nu})(\nu_0\sigma_0^2 + (\mathbf{y}-\mathbf{X}\widehat{\boldsymbol{\beta}})^t(\mathbf{y}-\mathbf{X}\widehat{\boldsymbol{\beta}}) + \\
&\quad (\boldsymbol{m}_0-\widehat{\boldsymbol{\beta}})^t(\boldsymbol{\Sigma}_0+(\mathbf{X}^t\mathbf{X})^{-1})^{-1}(\boldsymbol{m}_0-\widehat{\boldsymbol{\beta}}))\text{post. } \textbf{scale} \text{ (squared)} \\
\boldsymbol{W} &= (\boldsymbol{\Sigma}_0^{-1}+\mathrm{Var}(\widehat{\boldsymbol{\beta}})^{-1})^{-1}\mathrm{Var}(\widehat{\boldsymbol{\beta}})^{-1}
\end{aligned}$$

- See [Wak13, Expr. (5.46) & (5.47)] for these full conditionals with a change of notation and use of inverse gamma instead of our inv-χ^2.
- Very often, we see the **particular prior** with $\boldsymbol{m}_0 = \mathbf{0}$ and $\boldsymbol{\Sigma}_0 = \sigma_\beta^2\boldsymbol{I}$, i.e.,
$$[\boldsymbol{\beta}] = \mathrm{N}(\mathbf{0}, \sigma_\beta^2\boldsymbol{I}),$$
with the same inv-χ^2 prior as just given above.
- In this **particular case**, we have the **full conditional**,
$$[\boldsymbol{\beta} \,|\, \sigma^2, \mathbf{y}] = \mathrm{N}(\widehat{\boldsymbol{m}}, \sigma^2\widehat{\boldsymbol{\Sigma}})$$
where, now,
$$\begin{aligned}
\widehat{\boldsymbol{m}} &= (\boldsymbol{I}-\boldsymbol{W})\boldsymbol{m}_0 + \boldsymbol{W}\widehat{\boldsymbol{\beta}} \quad \text{cond. post. } \textbf{mean}, \\
&= \left(\frac{\sigma^2}{\sigma_\beta^2}\boldsymbol{I} + \mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^tBy \\
\sigma^2\widehat{\boldsymbol{\Sigma}} &= \boldsymbol{W}\mathrm{Var}(\widehat{\boldsymbol{\beta}}) \quad \text{cond. post. } \textbf{variance}, \\
&= \sigma^2\left(\frac{\sigma^2}{\sigma_\beta^2}\boldsymbol{I} + \mathbf{X}^t\mathbf{X}\right)^{-1} \\
\boldsymbol{W} &= (\sigma_\beta^{-2}\boldsymbol{I}+\mathrm{Var}(\widehat{\boldsymbol{\beta}})^{-1})^{-1}\mathrm{Var}(\widehat{\boldsymbol{\beta}})^{-1}
\end{aligned}$$

and the other full conditional posterior parameters remain the same as given in the general case of the independence prior, above.

- This gets us very close to typical (Bayesian) **shrinkage methods** and related methods, e.g., LASSO and ridge regression ([HTF01, §3.4], [Wak13, §10.5 & 10.6], [JWHT14, §6.2]), as well as to typical **Bayesian variable selection** methods (ref?).
- We never did answer the question, 'Now what?' That is, how does knowing the **full conditionals** get us closer to the posterior (and posterior predictive)? There are many answers to those questions, the most popular answer being **Gibbs sampling** or, more generally, Markov chain Monte Carlo (MCMC) methods (McMC?...). See [Wak13, §3.8] and INF511Notes.pdf Chapter 8 for a 2-stage Gibbs sampler algorithm that uses the full conditionals arising from a certain independence prior using the prostate data. In that same chapter, as an alternative to Gibbs sampling, we use **Hamiltonian Monte Carlo (HMC)** as implemented in **Stan**. We will wait to illustrate Gibbs sampling and Stan when we use an independence prior in the context of shrinkage methods, in a subsequent chapter of notes.

4.11 Other Priors

Generally speaking, for other priors on the parameters of the normal linear model, we will typically not recognize the posterior or full conditionals, in which case we have to appeal to other methods, MCMC again being being very popular, e.g., (Metropolis-)Hastings. See [Wak13, §3.8].

Ideally, we should choose a prior that reflects our prior beliefs about parameters; MCMC and other methods are well-developed and often allow us to specify priors without a concern for (conditional) conjugacy.

4.12 Summary II & Looking Ahead

We saw that the conjugate prior and the improper prior result in posteriors that are of known, standard form. In these cases, we simply have to somehow summarize the posterior, as illustrated for the improper prior case in INF511Notes.pdf §8.7.

As we will see in a subsequent chapter, Zellner's g-prior may be seen as a sort of combination of the conjugate prior's conditional prior for $\boldsymbol{\beta}$ and the improper prior's marginal for σ^2. And, like those two prior cases, we get a closed form of posterior (normal-inv-χ^2), if we are given g, i.e., conditional on g. In Bayesian model averaging/selection, we typically place a prior on g, as well as priors on different models from which we want to select/average. In this case, we no longer get a known form of posterior, but we do still get a known form of full conditional distribution for $\boldsymbol{\beta}$ and σ^2, conditional on g and on a model. This sets us up nicely to illustrate use of known full conditionals (for "Gibbs steps") as well as the use of an unknown full-conditional for g (for "Metropolis(-Hastings)" steps) inside of an iterative MCMC algorithm to sample parameters or unobserved outputs from their posterior (model full conditionals will be used in Gibbs steps).

Also, in a subsequent chapter introducing shrinkage, we well use a particular version of the independence prior, thus, we will have an opportunity to use the full conditionals for $\boldsymbol{\beta}$ and σ^2, now conditional on a "shrinkage parameter" (for "Gibbs steps"), along with a non-standard full conditional distribution of the shrinkage parameter (for "MH steps") in another implementation of MCMC for sampling from their posterior.

Given the close connection of these subsequent opportunities to the material here, we move ahead and wait to illustrate the implementation of the current material.

Lecture References

[Ber85] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, 2nd edition, 1985. ISBN 0387960988.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[Bre01] Leo Breiman. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statist. Sci.*, 16(3):199–231, 08 2001.

[Don17] David Donoho. 50 years of data science. *Journal of Computational and Graphical Statistics*, 26(4):745–766, 2017.

[GCS+14] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, Boca Raton, 3rd edition, 2014.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[JWHT14] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[KNNL05] Michael H. Kutner, Christopher J. Nachtsheim, John Neter, and William Li. *Applied Linear Statistical Models*. McGraw–Hill/Irwin, New York, 5th edition, 2005.

[Mur12] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[Rob01] Christian P. Robert. *The Bayesian Choice: From Decision–Theoretic Foundations to Computational Implementation*. Springer, New York, 2001. ISBN 0–387–95231–4.

[RS13] Fred L. Ramsey and Daniel W. Schafer. *The Statistical Sleuth: A Course in Methods of Data Analysis*. Brooks Cole, Boston, 3rd edition, 2013.

[RW06] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

[Wak13] Jon Wakefield. *Bayesian and Frequentist Regression Methods*. Springer, New York, 2013.